A Stealth Address-Powered Private and Composable Account Layer for Ethereum

Night Labs Team

25 October 2023

V1.0

Abstract

The Night Protocol whitepaper presents a pioneering stealth address-based privacy solution for Ethereum. It adeptly addresses the privacy paradox in blockchains, where transparency for accountability compromises user privacy. Night Protocol integrates stealth addresses with privacy pools and multi-party computation to offer robust anonymity while meeting compliance requirements. It highlights deficiencies in existing privacy solutions like computational overhead, scope limitations, and compliance challenges. The protocol capitalizes on elliptic curve cryptography and optimized dual-key stealth address schemes. Key innovations include Secp256k1 implementation and view tags for efficient fund scanning, paymaster architecture for abstracted gas fees, and multi-party threshold decryption for conditional anonymity. Salient architectural components include an intuitive front-end interface, SDK for dApp integration, Night Snap for easy MetaMask integration, and a private smart contract wallet. Diverse use cases are covered, including confidential payments, P2P transactions, private dApp interactions, and applications in healthcare. Quantum-resistant security and protection against denial-of-service attacks are analyzed as pivotal security considerations. Proposed future improvements encompass quantum-resistant cryptography, efficient STARK usage, enhanced privacy features, and integration across the Ethereum ecosystem. Overall, Night Protocol provides groundbreaking capabilities in delivering anonymity along with regulatory compliance on public blockchains. The well-rounded technical coverage and insights into design choices make this whitepaper a valuable resource for advancing research and development of privacy solutions.

Table of Contents

1. Introduction

The advent of blockchain technology has heralded a new era of digital transactions, offering unparalleled transparency and decentralization. However, this same openness has inadvertently cast a spotlight on a crucial concern: privacy. Addressing this concern forms the cornerstone of the Night Protocol, a revolutionary privacy solution for the Ethereum blockchain that harnesses the power of stealth addresses. This whitepaper delves into the workings and potential of Night Protocol, aiming to redefine privacy norms in the blockchain space.

1.1 The Privacy Paradox in Blockchain Technology

Blockchain technology, with its decentralized nature and transparent recording of transactions, has transformed our understanding of digital financial interactions. Yet, this very transparency has sparked a debate around privacy. Ethereum, a brainchild of Vitalik Buterin, is a prime example of this dichotomy. As an open, transparent ledger, Ethereum ensures accountability but also exposes users to potential privacy violations (*Buterin, 2023*).

Buterin himself has underscored the need for privacy within the Ethereum ecosystem, stating that "total privacy is not merely a personal right, but a security property that is vital for the safe and effective operation of decentralized applications." He further warns that without adequate privacy measures, visible data on the blockchain can be exploited by malicious entities, posing significant security risks (*Buterin, 2023*).

The challenge, therefore, lies in striking a balance between transparency and privacy in the blockchain domain. While absolute transparency can lead to surveillance and discrimination, total privacy could pave the way for illicit activities, undermining the trust blockchain technology seeks to establish.

1.2 Existing Privacy Solutions and Their Limitations

Several solutions, such as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs)—a form of zero-knowledge cryptography—have emerged to address these privacy issues. However, these solutions often involve complex calculations, restricting scalability and practical application on a large scale.

- **Trusted Setup Requirement:** Both Zcash (Zcash, 2014) and Tornado Cash (Pertsev et al., 2019) systems require a trusted setup. Zcash uses a ceremony to generate the initial parameters, which if compromised, could lead to the creation of false proofs, posing serious privacy and security risks. Tornado Cash also requires a ceremony to generate the initial parameters for zk-SNARKs.

- **Computationally Intensive:** zk-SNARKs, used in both Zcash and Tornado Cash, are computationally heavy, especially when creating the proofs. This can lead to scalability

issues and longer processing times, limiting the speed and number of transactions that can be conducted.

- **Limited Privacy Scope:** While Zcash provides the option to use shielded transactions that hide the sender, recipient, and value of a transaction, it still allows for transparent transactions, which can reveal transaction data. This means that the privacy provided isn't comprehensive and depends on the user's choice to use shielded transactions.

- **Privacy vs. Compliance Challenge:** In the case of Tornado Cash, despite the privacy provided by zk-SNARKs, it still faced issues with regulatory compliance. The protocol was used to launder illicit funds, leading to its inclusion on a sanctions list. This highlights the challenge of balancing privacy with the need for regulatory oversight and compliance.

- **Interoperability Issues:** zk-SNARKs require specific cryptographic support, which may not be available or easy to implement in all systems. This can limit their applicability and interoperability across different systems or blockchains.

- **Partial Anonymity:** While Tornado Cash uses zk-SNARKs to break the link between deposit and withdrawal addresses to provide transaction privacy, the deposits and withdrawals themselves are still visible on the blockchain, providing only partial anonymity.

1.3 Stealth Addresses: An Overview

Stealth addresses enhance confidentiality in transactions on public blockchains, where every transaction is transparently recorded. These addresses allow users to generate a unique, one-time address for each transaction, protecting the recipient's identity. Stealth addresses have found applications in various peer-to-peer interactions where privacy is paramount.

Historically, stealth addresses were first introduced in the Bittoken ecosystem and have been refined since. Notably, Nicolas van Saberhagen described the CryptoNote protocol (Saberhagen, 2013), which used stealth addresses to enhance blockchain transaction privacy. Following this, Peter Todd improved on the concept (Todd, 2014). Stealth addresses were later integrated into the Monero blockchain (Monero, 2014).

Night Protocol aims to fill this gap with a practical, scalable solution that robustly safeguards privacy on the Ethereum blockchain. It employs stealth address technology to create private, composable accounts, enabling users to engage with popular Ethereum decentralized applications (dApps) and conduct transactions while preserving their privacy.

1.4 The Promise of Night Protocol

Night Protocol is an open stealth address protocol that can offer stealth addresses on the application layer of programmable blockchains like Ethereum. Night Protocol is designed to be fully extendable, enabling the creation of unique stealth address protocols based on specific cryptographic schemes.

Our proposed fundamental protocol is indifferent to different cryptographic frameworks and has the ability to significantly enhance user interactions with stealth addresses within the context of programmable blockchains. Night Protocol embodies a primary implementation that incorporates the necessary reusable capabilities for any trustless stealth address protocol. Furthermore, Night Protocol is designed to accommodate future quantum-resistant cryptographic schemes that require larger key sizes.

Through this whitepaper, we will delve into how Night Protocol plans to revolutionize privacy norms on the Ethereum blockchain, offering a comprehensive solution that ensures both the security and privacy of users. The key contributions of this work are as follows:

- We have engineered an initial prototype for stealth addresses that capitalizes on the Secp256k1 elliptic curve, demonstrating superior performance in relation to parsing time as compared to existing Stealth Address Protocols (Wahrstatter and Solomon, 2023).

- While privacy is certainly paramount, it only attains significance and value within the framework of compliance. Drawing from the central ideas of Buterin's (2023) work on Privacy Pools — a groundbreaking privacy-enhancement protocol based on smart contracts. We employ secure multiparty computation (MPC) for encrypting and decrypting user transactions and addresses. This approach safeguards user privacy while concurrently satisfying regulatory requirements.

- Gas Abstraction is a technique devised to tackle transactions devoid of Gas and to facilitate the payment of Gas via ERC20 tokens. The Night Protocol proposes two potential implementation strategies for Gas Abstraction: Paymaster (EIP 4337) and Relayer. Through our proposed Gas Abstraction, we have actualized Gas-less experiences, catering to the needs of NFT and other ERC20 token transfers.

- We underscore the inherent modularity of our protocol, emphasizing the considerable potential of such methodologies when incorporated at the application layer of programmable blockchains. This could confer advantages to a variety of domains including, but not limited to, Smart Contract wallets, token donation platforms, funding for public goods, decentralized finance, and the Non-Fungible Token ecosystem.

With Night Protocol, we aim to shift the paradigm in blockchain privacy, paving the way for a new era of secure and private digital transactions.

2. Background and related work

2.1 Introduction

Stealth addresses are fundamental for ensuring privacy in blockchain transactions. This review traces the evolution and application of stealth addresses, highlighting significant contributions and advancements in the field.

2.2 Early Developments and Fundamental Protocols

Bytecoin (2011) pioneered the introduction of stealth addresses in blockchain, marking a crucial step towards anonymous transactions. Van Saberhagen (2013) and Todd (2014) later refined these initial concepts, providing a more robust framework that eventually led to the development of the Double-Key Stealth Address Protocol (DKSAP) in Monero in 2014.

Courtois and Mercer (2017) not only traced the development history of stealth addresses but also enhanced the DKSAP by introducing multiple spending keys. This innovation increased resistance to attacks but also introduced the complexity of managing multiple keys for users, a trade-off that future researchers needed to address.

2.3 Protocol Enhancements and Efficiency Improvements

Fan (2018) made significant strides in improving the efficiency of DKSAP. By allowing sender-receiver pairs to reuse their generated Diffie Hellman secret with an increasing counter, Fan's approach facilitated faster parsing and achieved a notable 50% performance improvement compared to the standard DKSAP, making it a pivotal work in the field for its contribution to efficiency.

Fan et al. (2019) further streamlined the DKSAP by reducing the number of keys required from two to one, a crucial improvement that not only maintained the protocol's properties but also reduced storage requirements, addressing a significant challenge in key management and storage in stealth addresses.

Feng et al. (2020, 2021) introduced protocols that were not only efficient but also secure, with PDKSAP particularly focusing on preventing temporary key leakage, addressing a critical security concern in stealth address transactions.

Liu et al. (2019) integrated stealth addresses with ring signatures, creating a confidential layer within a token system. This integration not only provided enhanced privacy but also shielded transaction information, marking a significant advancement in the application of stealth addresses for transaction privacy.

Wahrstätter (2023) proposed and analyzed stealth address schemes and protocols that were integral to privacy in blockchain transactions, providing valuable insights and frameworks for future research and application in the field.

2.4 Applications in Healthcare and Real Estate

Lee and Song (2021) applied stealth address protocols and ring signatures to facilitate confidential transactions on a private Ethereum network, with a focus on healthcare information exchange. Their work is noteworthy for its application of stealth addresses in healthcare, demonstrating the protocol's versatility and potential in various domains.

Mohideen and Kumar (2022) extended the application of stealth addresses to the real estate sector, emphasizing privacy-preserving data transfer. Their work is significant for

showcasing the practicality and benefits of stealth addresses in real estate transactions, opening avenues for further research and application in the sector.

The literature on stealth addresses has expanded over the years, with researchers making continuous improvements and innovations. The body of work underscores the importance of stealth addresses in ensuring privacy in blockchain transactions, highlighting the ongoing efforts to enhance their efficiency, security, and application in various domains.

3. Stealth Address Technology and Cryptographic Principles

This section will dive deeper into the cryptographic principles that underpin stealth address technology and provide a detailed step-by-step explanation of how stealth addresses are generated and used within the Night Protocol.

3.1 Elliptic Curve Cryptography

The generation and use of stealth addresses in the Night Protocol depend on fundamental cryptographic principles: Elliptic Curve Cryptography (ECC). ECC is a form of public key cryptography that uses the mathematics behind elliptic curves to generate secure cryptographic keys. The ECC allows for strong security with relatively small key sizes, making it an efficient choice for systems that need to conserve resources.

We define an elliptic curve E over a finite field F_p, with p being a 256-bit prime, represented in Weierstrass form as $y^2 = x^3 + ax + b$ where $x, y \in F_p$, and a and b are constants. Points (x, y) on E form an Abelian group, allowing point addition operation $R = P + Q$ for any $P, Q \in E$.

Scalars, denoted by lowercase letters, are random n-bit integers $(p, q \in \{0, 1\}^n)$, commonly of 256 bits as in the Secp256k1 curve. EC multiplication involves repeated point additions, $n \times P = \sum_i^{i+n} P_i$, with the operation being commutative. The generator point G facilitates public key derivation as $P = p \times G$.

The "point at infinity" O serves as the identity in EC arithmetic, with $O + O = O$ and $P + O = P$. Each point P has an inverse, satisfying $(-P) + P = O$.

The Standards for Efficient Cryptography (SEC) proposes standardized elliptic curves for cryptography, including the well-known Secp256k1, defined by $y^2 = x^3 + 7 \pmod p$, with $p = 2^{256} - 2^{32} - 977$. Secp256k1, utilized in Bittoken's ECDSA, offers a large prime order and efficient arithmetic, making it ideal for various applications, including cryptocurrencies, blockchain, IoT, and secure communications.

Our stealth address protocol is elliptic curve-agnostic, with the initial implementation using the Secp256k1 curve.

3.2 ISAP and DKSAP

The following section is divided into different parts. First, we explicate how users generate a stealth address using the Improved Stealth Address Protocol (ISAP), described by Todd(2014), when they want to perform a transaction. Second, we introduce the DKSAP, described by van Saberhagen(2013), primarily focusing on the parsing process which the receiver or third-party providers can perform.

3.2.1 ISAP

3.2.1.1 Stealth Address Generation

In the context of stealth address generation, consider two autonomous entities: a sender, denoted as C, and a recipient, denoted as R. Each party possesses a unique cryptographic key pair: C has (p, P) and R has (r, R). The recipient's public key, R, is assumed to be public knowledge, accessible to the sender.

Notably, for each transaction initiated under the stealth address protocol, the sender generates a transient ephemeral key pair, (p, P), unlinked to their identity. This practice enhances the privacy and security of the transaction process.

The generation process unfolds as follows:

1. Ephemeral Key Generation: C produces and publicizes an ephemeral key pair (p, P).

2. DH Secret Creation: The ephemeral private key is multiplied by R, yielding a DH secret k, which can also be expressed as $k = r \times P = p \times R = r \times p \times G$.

3. Hashing: The shared secret k undergoes a hashing process, represented by the function $h: X \to Y$, resulting in $h(k)$.

4. Generator Point Multiplication: The hashed secret is then multiplied by a generator point G, producing $K_h = h(k) \times G$.

5. Stealth Address Derivation: The outcome of the fourth step is added to R, deriving the stealth address $R_{st} = K_h + R$.

The derived stealth address, R_{st}, is utilized by C for transactions with R, ensuring anonymity as there is no discernible link between the two parties observable by external entities.

3.2.1.2 Stealth Address Parsing

The parsing process enables recipients to identify and access their stealth addresses. This involves parsing through all published ephemeral public keys, collectively represented as $A = \{Pi, \ldots, Pn\}$.

For each $P \in A$, the recipient performs the following steps:

1. Shared Secret Derivation: Multiply P by the private key r to obtain $k = r \times P$.

2. Hashing: The derived shared secret k is hashed to obtain $h(k)$.

3. Private Key Adjustment: The hashed secret is added to the private key r, resulting in $r_{st} = h_k + r$.

4. Stealth Public Key Derivation: Multiply r_{st} by G to derive the stealth public key $R_{st} = r_{st} \times G$.

5. Address Derivation: Hash Rst*Rst*and extract the least significant 20 bytes to obtain the address $R_{st}^{addr} = h(R_{st})[-20:]$.

Upon successful derivation, the recipient can verify if R_{st}^{addr} has engaged in transactions or received assets, subsequently storing the private key r_{st} if verification is successful.

In essence, the protocol exploits the mathematical property $k_h \times G + P = (k_h + p) \times G$, facilitating the derivation of a stealth address through divergent paths, with the private key for the stealth address exclusively generable by the recipient.

3.2.2 Dual-Key Scheme (DKSAP)

Dual-Key Stealth Address Protocol (DKSAP), a mechanism enhancing both user experience and transaction security. DKSAP, an advanced extension of the Improved Stealth Address Protocol (ISAP), introduces an auxiliary key pair designated exclusively for the parsing process. Under DKSAP, recipients are endowed with two distinct key pairs: the scanning keys (r_{SC}, R_{SC}) and the spending keys (r_{SP}, R_{SP}).

This dual-key architecture allows for a strategic separation between the scanning key pair, still integral to the generation of the Diffie-Hellman (DH) secret, and the process of stealth address generation. Implementing DKSAP involves the following steps executed by the sender:

1. Ephemeral Private Key Multiplication: The sender multiplies the randomly generated ephemeral private key with the recipient's scanning public key: $k = p \times R_{SC}$.

2. Shared Secret Hashing: The shared secret k is hashed, and the resultant hash $h(k)$ is multiplied by the generator point G, yielding $K_h = h(k) \times G$.

3. Stealth Address Generation: The outcome of step 2 is added to the recipient's spending public key to derive the stealth address: $R_{st} = K_h + R_{SP}$.

Post this process, recipients have two alternative methodologies for identifying their corresponding stealth address R_{st}. The first approach involves calculating the DH secret by multiplying the scanning private key r_{SC} with the ephemeral public key P. With the DH secret in possession, the recipient can hash it, multiply the hash with the generator point, and add the result to the spending public key to derive the stealth address. Alternatively, the recipient can add the DH secret to the spending private key and multiply the resultant

sum with the generator point to obtain the stealth address: $R_{SP} + h(r_{SC} \times P) \times G = (r_{SP} + h(r_{SC} \times P)) \times G$.

A pivotal feature of DKSAP is the provision allowing recipients to securely share their scanning private key r_{SC} with third-party parsing providers without jeopardizing the integrity of the spending private key. This mechanism enables third-party services to undertake the parsing process and alert users of incoming transactions to their stealth addresses. However, these third-party entities can not access stealth addresses without the spending private key r_{SP}, ensuring the assets remain secure and inaccessible.

4. Night Protocol - a stealth address-powered private account layer

4.1 High level Architecture

Night represents a pioneering privacy protocol solution, ingeniously integrating Stealth Address and Privacy Pools technologies. This innovative amalgamation aims to resolve pivotal challenges endemic to the blockchain domain, including privacy, compliance, and traceability. Night is distinctive due to its dual functionality: it not only efficaciously shields users' privacy on the blockchain—protecting sensitive data such as asset quantities and transaction behaviors — but also offers a holistic solution for digital asset transactions through the incorporation of compliance tools.

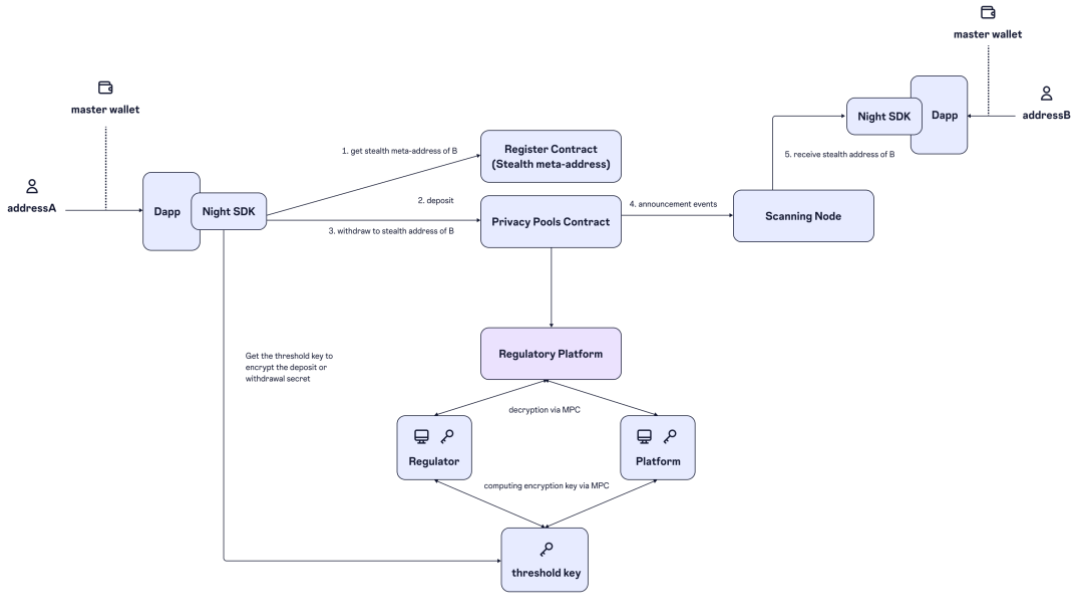


Figure 1. Night protocol high level architecture

4.1.1 Privacy Protection Mechanism

4.1.1.1 Stealth Address

A cornerstone of Night's privacy-preserving features is its implementation of stealth address technology. This technology is instrumental in concealing vital transaction data, including participant identities and asset amounts involved in transactions. Stealth addresses engender an environment where users can confidently execute transactions on the blockchain, alleviating concerns regarding privacy breaches.

4.1.1.2 Privacy Pools

In addition to stealth addresses, Night introduces Privacy Pools, a mechanism designed to augment privacy protection levels. Privacy Pools achieve this by amalgamating transactions from multiple users into a singular pool. This innovative approach effectively obfuscates the identities of transaction initiators, thereby complicating the process of transaction traceability on the blockchain.

4.1.2 Integration and Compatibility

Night is meticulously designed to ensure seamless integration with prevailing wallet standards and protocols in the blockchain ecosystem. This compatibility feature facilitates the easy adoption of Night, allowing users to effortlessly integrate it into their existing digital wallet infrastructure. Such interoperability is vital for fostering Night's widespread adoption and for providing users with convenient privacy protection tools, especially when interacting with Web3 DApps.

4.1.3 Compliance and Regulatory Adherence

A salient feature of Night is its commitment to compliance and adherence to regulatory standards, which is indispensable for the sustainable growth and development of the digital asset sector. Night is equipped with features that not only fortify user privacy but also meet the stringent requirements imposed by governmental bodies and financial institutions. This dual focus on privacy and compliance positions Night as a comprehensive privacy protocol solution adept at satisfying the privacy requisites of individual users while simultaneously aligning with legal and regulatory mandates.

4.1.4 User Journey

4.1.4.1 Setup account

This part delineates a systematic procedure for generating a Stealth meta-address and registering it within the Register Contract. The process is meticulously segmented into four

pivotal steps, each contributing to the creation and registration of a Stealth meta-address, a crucial component for ensuring privacy in transactions.

Step 1: Message Signing and Signature Retrieval

Initiate the process by crafting a message utilizing a wallet, followed by signing this message. The resultant signature from this step is pivotal as it acts as the seed for the subsequent stages of the process.

Step 2: Generation of Key Pairs: spend_key and view_key

With the obtained signature serving as the seed, generate two integral key pairs: spend_key and view_key. These keys play a crucial role in the generation of the Stealth meta-address.

- spend_key (sk_{spend}, PK_{spend}): The spend_key comprises the spending key and its corresponding public key. It is indispensable for executing transactions and authenticating fund expenditures.
- view_key (sk_{view}, PK_{view}): The view_key, consisting of the viewing key and its associated public key, is utilized for observing the balance and transaction history. However, it lacks the capability to expend funds.

Step 3: Stealth Meta-Address Generation

Upon the acquisition of spend_key and view_key, amalgamate these keys to engender the Stealth meta-address. The format of this address is delineated as: 0x<PK_{spend}>< PK_{view}>. The Stealth meta-address is instrumental in safeguarding user privacy as it obfuscates the direct link between the user's identity and transaction history.

Step 4: Register Contract: Stealth Meta-Address Registration

Subsequent to the generation of the Stealth meta-address, proceed to register this address within the Register Contract. This registration ensures the association of the Stealth meta-address with the user's account, rendering it operational for transactions. The registration phase may necessitate linking the Stealth meta-address with the user's wallet or identity data, facilitating the smart contract in recognizing and processing transaction requests efficiently.

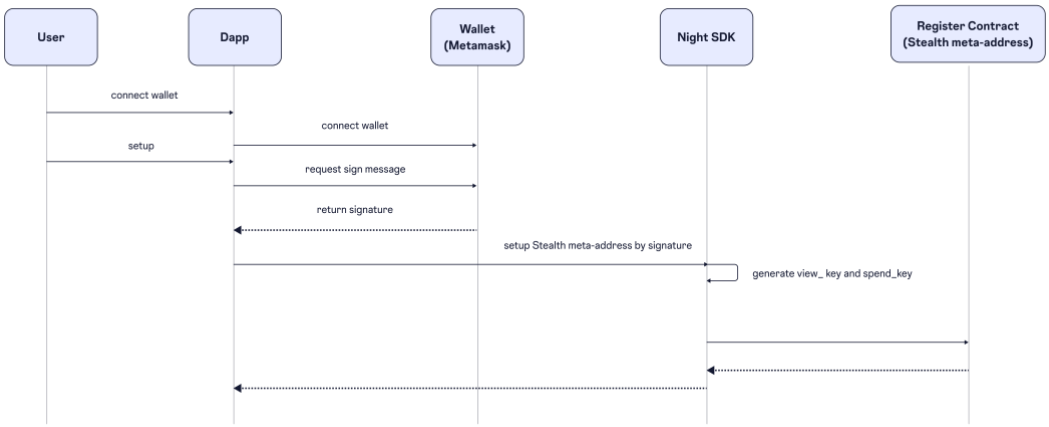


Figure 2. Stealth address setup

4.1.4.2 Re-sign procedure

Upon successful registration of a wallet equipped with Stealth meta-address functionality, a user, when necessitating subsequent access to said wallet, is mandated to navigate through the designated re-sign procedure intrinsic to the stealth address. The ensuing delineation provides a meticulous walkthrough of the requisite steps inherent to this re-sign process:

- Initiation via Message Signing: Initially, the user is compelled to craft and duly sign a message utilizing their respective wallet. The signature thus procured from this preliminary message serves a pivotal role, acting as the seed indispensable for the ensuing generation of cryptographic keys.

- Key Generation - spend_key & view_key: Subsequent to obtaining the message signature, the user employs this signature as a basis to generate two crucial key pairs: the spend_key (denoted as sk_{spend}, PK_{spend}) and the view_key (represented as sk_{view}, PK_{view}).

- Stealth Meta-Address Construction: Armed with the spend_key and view_key, the user is now positioned to construct their unique Stealth meta-address. This address, typically formulated as: 0x<PK_{spend}>< PK_{view}>, serves as an identifier for the user's wallet, thereby facilitating the conduct of transactions and granting access to the assets contained within the wallet.

- Re-sign Confirmation: Upon the successful formulation of the Stealth meta-address, users are enabled to utilize this address as a means to log into their wallet. This meticulously designed process not only ascertains the user's identity with precision but also guarantees their exclusive access rights. Consequently, users are endowed with the capabilities to manage their digital assets securely and execute transactions with confidence and ease.

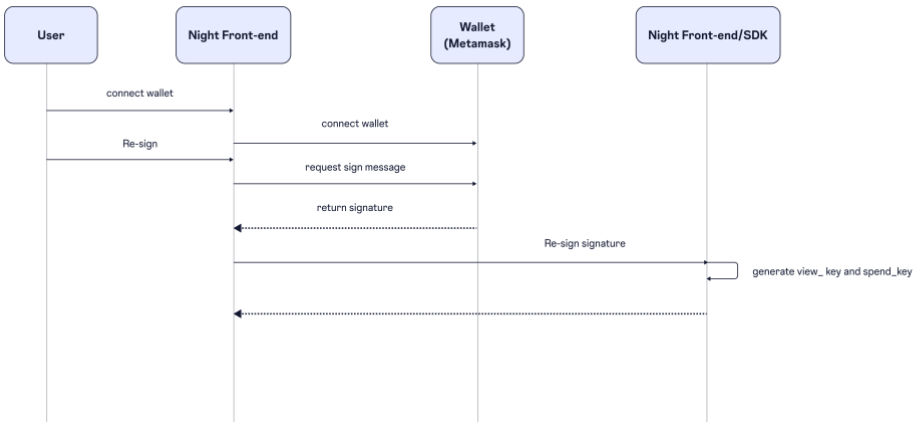


Figure 3. Stealth address re-sign

4.1.4.3 Deposit funds to privacy pools

To facilitate the transfer of funds from a user's primary (master) wallet to a stealth address, adhere to the following sequential steps:

- Deposit message construction: Initially, craft a deposit message encompassing various elements: commitment, token, amount, ciphertext, and trace_secret.
  - Random secret generation: Commence by generating a random secret.
  - Commitment calculation: Compute the commitment using the formula $commitment = H(secret, token, amount) commitment = H(secret, token, amount)$. This commitment symbolizes the deposit commitment.
  - Ciphertext generation: Employ the function $aes_enc_hmac(secret)$to generate ciphertext, which will be pivotal in creating the deposit proof during withdrawal processes.
  - Trace_Secret construction: Utilize the function $threshold_enc(secret, time)$ to construct trace_secret. This element, crucial for tracing fund destinations, is encrypted via threshold encryption and necessitates collaborative decryption by multiple parties (known as MPC-threshold-decryption).
- Master address signature: Proceed to sign the meticulously constructed deposit message utilizing the master address.
- Deposit transaction initiation: Subsequently, initiate a deposit transaction.
- Stealth address generation: Engage in the generation of stealth address.
- Withdrawal message construction: Construct a withdrawal message incorporating various components: proof, root, nullifier, token, amount, recipient, relayer, fee, and trace_secret.

- Proof: The deposit proof, with private inputs comprising the secret and merkle_branch, and public inputs including the nullifier and merkle_root.
- Root: The root of the Merkle tree.
- Nullifier: A mechanism to prevent the occurrence of double-spending.
- Recipient: The stealth address of the recipient.
- Relayer: The address of the wallet contributing gas.
- Fee: The contribution towards gas.
- Trace_Secret Construction: Similar to the deposit message, use $threshold_enc(secret, time)$for trace_secret, essential for tracing fund destinations and requiring MPC-threshold-decryption.

- Withdrawal Transaction Initiation: Finally, initiate a withdrawal transaction to complete the process.

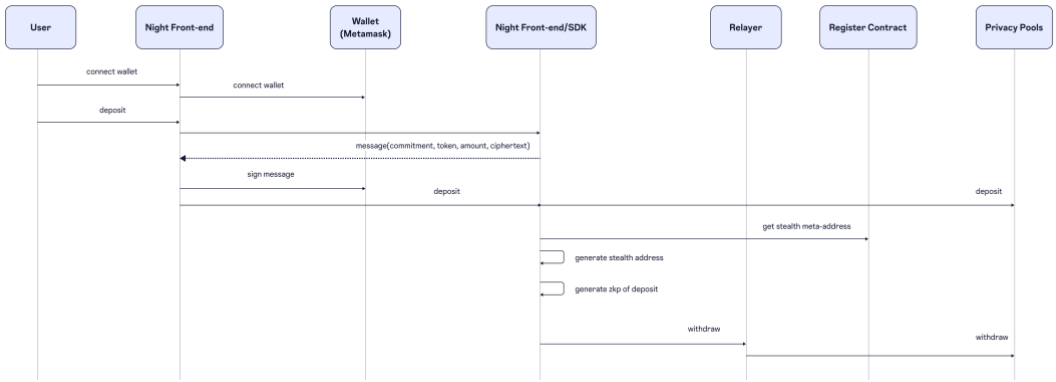


Figure 4. Deposit funds to privacy pools

4.1.4.4 Transfer funds

There are two types of transfers available: one is conducted through the Master address, applicable when a user has not yet created a stealth address; the other is executed through the stealth address, which is relevant when a user has already established a stealth address and there are funds present under this address.

The procedure for transferring funds through a master address and stealth address is delineated below:

- **Deposit message formulation:**
  - Random secret generation: Initiate the process by generating a random secret.
  - Commitment construction: Calculate the commitment using the equation $commitment = H(secret, token, amount)$. This commitment serves as a representation of the deposit commitment.

- Ciphertext formulation: Employ $aes_enc_hmac(secret)$to formulate the ciphertext, which is instrumental in generating the deposit proof during the withdrawal phase.
- Trace_Secret creation: Implement $threshold_enc(secret, time)$to create trace_secret. This component is vital for tracing the destination of funds and is encrypted through threshold encryption, necessitating multi-party cooperation for decryption (known as MPC-threshold-decryption).

- **Signature:** Authenticate the deposit message by signing it with the master address when transferring the funds through a master address. Similarly, sign the deposit message with the stealth wallet when transferring the funds through a stealth address.
- **Deposit transaction Initiation:** Launch the deposit transaction.
- **Stealth address generation:** Proceed with the generation of the stealth address.
- **Withdrawal message formulation:** Craft a withdrawal message comprising various elements: proof, root, nullifier, token, amount, recipient, relayer, fee, and trace_secret.
  - Proof: The deposit proof with private inputs (secret, merkle_branch) and public inputs (nullifier, merkle_root).
  - Root: The Merkle tree root.
  - Nullifier: A mechanism integral for the prevention of double-spending.
  - Recipient: The stealth address of the intended recipient.
  - Relayer: The wallet address that contributes gas.
  - Fee: The amount contributed towards gas.
  - Trace_Secret creation: Similar to the deposit message, utilize $threshold_enc(secret, time)$to create trace_secret, essential for fund destination tracing and requiring MPC-threshold-decryption.
- **Withdrawal transaction initiation:** Initiate the withdrawal transaction.
- **Contract-Based fund withdrawal completion:** Finalize the fund withdrawal within the contract and subsequently publish the announcement event.
- **Receiver's stealth address detection:** The receiver engages in the calculation of the announcement event and successfully detects their own stealth address.

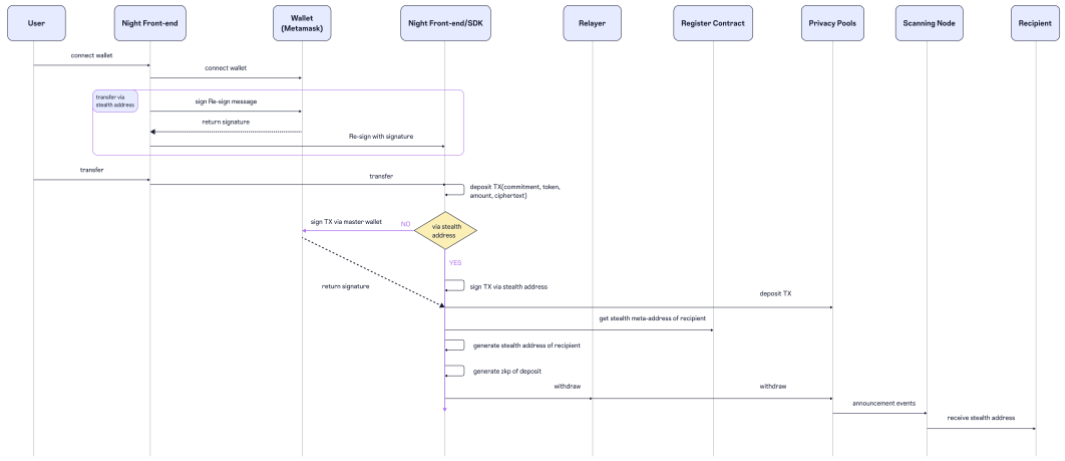


Figure 5. Transfer funds procedure

4.2 Support and Efficiency Optimization

4.2.1 Paying transaction fees

While stealth addresses offer anonymity, they still necessitate traditional gas fees for transaction processing, which can pose challenges for users. Meta-Transactions serve as the innovative answer to this gas issue within the Night Protocol.

Meta-transactions in the Night Protocol simplify the gas fee complexities associated with stealth addresses, making interactions more user-friendly. Here's how:

- **User Privacy:** Stealth addresses preserve user privacy by obfuscating transaction recipients. Meta-transactions retain this privacy while abstracting away gas fee concerns.
- **User Convenience:** Users can interact with the Night Protocol without needing to manage token balances solely for gas fees.
- **Enhanced Scalability:** As more users adopt the Night Protocol, the abstraction of gas fees ensures smooth scalability, as users won't be deterred by the complexities of managing gas fees.

The user flow is as follows.

Initiating a Stealth Address Transaction: The user begins by initiating a transaction within the Night Protocol. This could involve sending tokens, interacting with a smart contract, or any other protocol-specific action.

Creating a Meta-Transaction: Instead of directly sending the transaction to the blockchain network, the Night Protocol creates a *meta-transaction* on behalf of the user. This meta-transaction includes all the relevant transaction details, such as the recipient's stealth address, the amount to be sent, and the action to be performed.

Signing the Meta-Transaction: The Night Protocol prompts the user to sign the meta-transaction using their private key. This signature serves as proof of the user's intent and authorization for the transaction.

Identifying a Relayer: With the meta-transaction signed, the Night Protocol helps the user identify an appropriate *relayer* from its network. Users may have choices among relayers, each with its own fee structure and service level. Users can select a relayer that suits their preferences, such as cost-effectiveness or transaction speed.

Sending the Meta-Transaction to the Chosen Relayer: The user sends the signed meta-transaction to the selected relayer. This can be done through a user-friendly interface within the Night Protocol.

Relayer Processes the Meta-Transaction: The chosen relayer receives the meta-transaction and begins processing it. The relayer verifies the user's signature and ensures the transaction details are correct. Importantly, the relayer has the necessary token balance to cover the gas fee required for the transaction to be included in the blockchain.

Paying the Gas Fee: Using its token funds, the relayer pays the gas fee to the blockchain network on behalf of the user. This process is transparent to the user, and they are not required to have any tokens for gas fees in their wallet.

Transaction Submission: Once the gas fee is paid, the relayer submits the meta-transaction to the blockchain network.

Blockchain Confirmation: The blockchain network processes the transaction, validating the user's signature and executing the specified action, such as transferring tokens or interacting with a smart contract.

Transaction Confirmation: Once the transaction is confirmed on the blockchain, the Night Protocol provides the user with confirmation of the successful transaction.

End of Transaction: The user's interaction with the Night Protocol is complete. The user has successfully conducted a stealth address transaction without having to deal with the intricacies of gas fees.

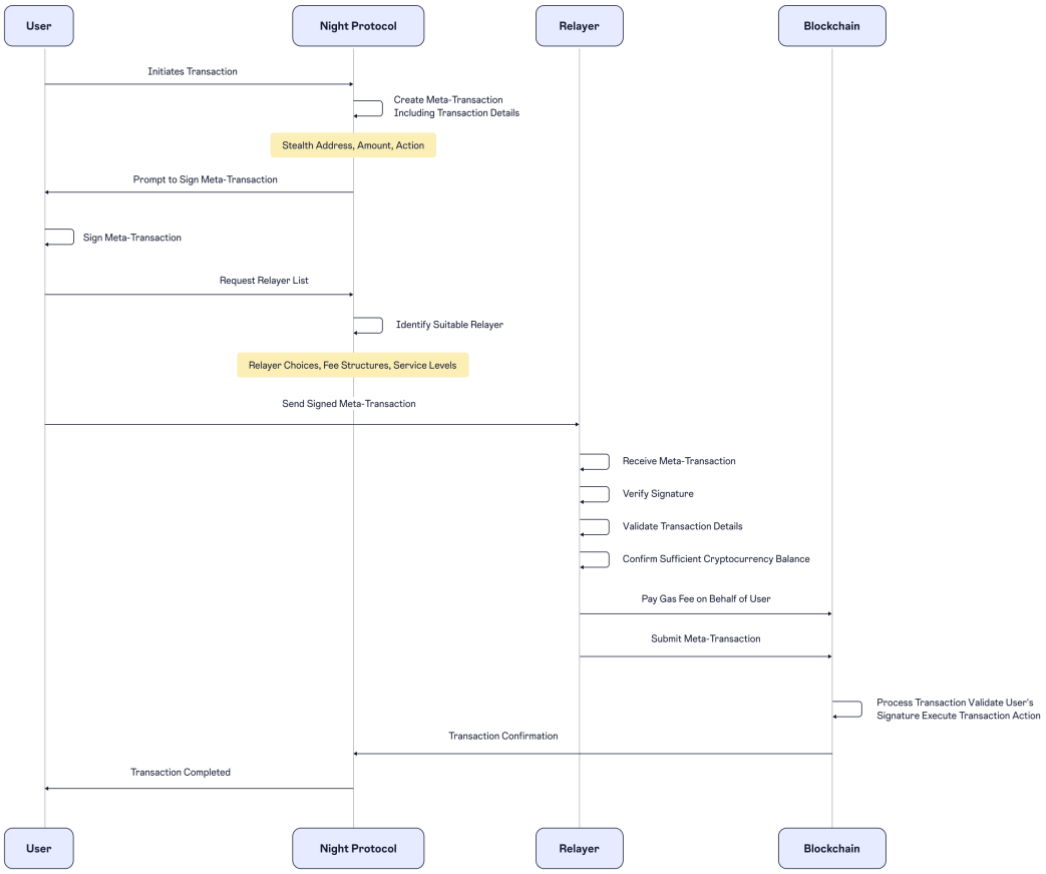


Figure 6. Pay transaction fees via Meta-Transaction

4.2.2 Fund scanning efficiency

Within Night Protocol, the optimization of fund scanning efficiency is a critical aspect that directly impacts user experience and the protocol's overall effectiveness. To achieve this, Night Protocol leverages a combination of Secp256k1 Implementation and View Tags, which work in harmony to streamline the fund scanning process. This section delves into the technical details of how these two components synergize to enhance efficiency.

4.2.2.1 Secp256k1 Implementation

One of the primary efficiency enhancements within Night Protocol is the integration of the Secp256k1 elliptic curve cryptography implementation. This cryptographic scheme is well-suited for blockchain environments, particularly Ethereum, and significantly improves the protocol's viability for practical implementation. Our analysis of existing protocols revealed two key deficiencies that hindered their real-world usability:

1. Parsing Process Efficiency: In the traditional approach, each potential recipient had to decode every announcement, which could be excessively time-consuming. Night Protocol addresses this by optimizing the parsing process, reducing the computational overhead required for recipients to identify themselves as intended recipients.

2. Insufficient Information in Announcements: The announcement, which typically contains the ephemeral public key, often lacked sufficient information for recipients to identify the relevant assets and rights in a stealth interaction. Night Protocol enhances the announcement process to provide recipients with more comprehensive information, making fund scanning more efficient and effective.

4.2.2.2 View Tags

View tags, a concept borrowed from the Monero blockchain protocol, play a pivotal role in further optimizing the fund scanning process within Night Protocol. View tags allow recipients to bypass certain steps in the parsing process under specific conditions, significantly reducing computational overhead. Here's how view tags work in synergy with Secp256k1 Implementation:

- View Tag Size and Significance: In Night Protocol, the size of the view tag (denoted as "n") can be kept very small, such as $n = 1$. This means that only a single byte of data needs to be processed for the view tag. This trade-off introduces computational efficiency while ensuring privacy. A smaller n implies that a full derivation of the stealth address needs to be attempted only $1/256$ of the time.

- Construction of View Tags: Senders use their ephemeral key pair (p, P) to compute the hashed Diffie-Hellman (DH) secret $\left(k_h = h(p \times R_{SC})\right)$ and select the most significant n bytes of k_h to construct the view tag, denoted as $Q(Q = k_h[:n])$. This view tag is shared alongside the stealth address transaction.

- Recipient Verification: Recipients, possessing the scanning key pair (r_{SC}, R_{SC}), follow a parallel procedure to compute their view tag $(Q' = h(r_{SC} \times P)[:n])$. They then compare this recipient-computed view tag (Q') to the view tag listed in the announcement (Q).

4.2.2.3 Streamlined Parsing Process

The parsing process within Night Protocol is significantly optimized through the seamless integration of Secp256k1 Implementation and View Tags. Here are the key steps involved when parsing each announcement $\left(P, R_{st_{addr}}, Q\right)$ within the set "a", where "a" belongs to the announcement set "A":

1. Elliptic Curve Multiplication: The recipient multiplies the ephemeral public key P with their scanning private key r_{SC} to obtain $k(k = r_{SC} \times P)$.

2. Hashing the Shared Secret: The derived shared secret k is then hashed to compute $k_h\left(k_h = h(k)\right)$.

3. View Tag Derivation: From the hashed shared secret k_h, the recipient derives their view tag $(Q' = k_h[:n])$.

4. Comparing View Tags: The recipient compares the derived view tag (Q') with the view tag in the received announcement (Q).

5. Stealth Address Verification: If the view tags match $(Q == Q')$, the recipient proceeds to compute the stealth address and compares it to the address logged in the announcement $\left({R_{st_{addr}}}' = h\left((k_h + r_{SP}) \times G\right)[-20:] = R_{st_{addr}}\right)$.

The harmonious integration of Secp256k1 Implementation and View Tags within Night Protocol significantly expedites the fund scanning process. This innovative approach reduces computational overhead, enhances user experience, and maintains robust privacy protections within the protocol. Ultimately, Night Protocol stands as a testament to the power of cryptographic optimization in achieving efficient and secure blockchain interactions.

4.2.2.4 Third-party scanning services

Scanning each individual Announcement can be a time-consuming task, potentially resulting in delays in identifying incoming funds. To address this issue and expedite the process, Night Protocol introduces the concept of third-party scanning services, offering users an efficient solution while preserving their fund security.

Here is how third-party scanning services can enhance fund scanning efficiency within the Night Protocol:

Dual-Key Setup: Night Protocol employs a dual-key setup, where each recipient possesses two private keys: sk_{spend} and sk_{view}. Correspondingly, they publish the associated public keys: PK_{spend} and PK_{view}.

Encryption and Stealth Address: When a sender intends to transfer funds to a recipient, they generate a random number r and encrypt it using the recipient's public view key PK_{view} and an ephemeral private key $sk_{ephemeral}$. This process produces a ciphertext c.

Stealth Address Generation: The sender computes the stealth address, represented as the address derived from $PK_{stealth} = PK_{spend} * r$. Subsequently, they send the funds to this generated stealth address.

Event Emission: Within Night Protocol, a contract emits the ciphertext c, ephemeral public key $PK_{ephemeral}$, and the stealth address $a\_stealth$ as part of the transaction event.

Recipient Verification: Upon receiving these events, the recipient utilizes their private view key sk_{view} and the provided ephemeral public key $PK_{ephemeral}$ to decrypt the random number r. The recipient can then check if $PK_{stealth} = PK_{spend} * r$, confirming that they indeed control the corresponding stealth address $a\_stealth$.

By following this approach, Night Protocol enables recipients to securely share their private view keys (sk_{view} and PK_{spend}) with third-party scanning services. These services can efficiently check whether the recipient has received funds without gaining access to the

ability to spend those funds. This delegation of scanning to trusted third-party entities helps optimize fund scanning efficiency while maintaining the highest level of fund security.

In summary, Night Protocol combines cryptographic security measures with the option of delegating fund scanning to third-party services to ensure that fund scanning is not only efficient but also secure for users. This approach provides a balance between user convenience and asset protection, enhancing the overall experience within the protocol.

In addition to the third-party scanning services, Night Protocol incorporates several optimizations to further enhance fund scanning efficiency, ensuring a smooth and responsive user experience:

Parallel Computing: Night Protocol harnesses the full potential of multi-core processors and distributed computing. By leveraging multi-threading and distributed computing, the protocol can process multiple fund scanning requests simultaneously. This approach significantly boosts system responsiveness, especially when dealing with a high volume of scanning requests. Parallel computing enables Night Protocol to efficiently handle increased transaction loads.

Caching Mechanism: Night Protocol implements a robust caching mechanism designed to eliminate redundant calculations. When the same address undergoes multiple scans, the protocol stores the results of these scans in a cache. Consequently, during subsequent scans of the same address, Night Protocol can retrieve and return the previously computed results without the need for recomputation. This strategy substantially reduces computational overhead and minimizes latency, ensuring speedy fund scanning.

Precomputation: In scenarios where it is feasible, Night Protocol conducts precomputation and stores precomputed results in a database. These precomputed results can be utilized during real-time fund scanning, reducing the computational demands of on-the-fly calculations. This practice effectively decreases system load and enhances overall performance, offering a more responsive fund scanning process.

These optimization strategies collectively contribute to Night Protocol's efficiency in scanning and verifying incoming funds, ensuring that users can enjoy a swift and seamless experience while maintaining the highest level of security. As the protocol evolves, these optimization measures will continue to be refined and expanded upon to meet the demands of an ever-growing user base and increasing transaction volumes.

5. Balancing Privacy and Compliance

In this section, we introduce privacy pools built upon Tornado Cash, with a core mission to provide robust privacy safeguards while addressing the multifaceted requirements of regulatory compliance.

5.1 The Tornado Cash Privacy Solution

Tornado Cash stands as a pioneering privacy protocol, founded on the fundamental concept of fund mixing to attain enhanced anonymity. While it offers an exceptional level of privacy protection, there emerge scenarios where regulatory compliance mandates the disclosure of transaction specifics. To better understand how Night Protocol integrates privacy and compliance, let's delve into the intricacies of Tornado Cash's privacy solution (Pertsev et al. 2019).

5.1.1 The Deposit Process

Tornado Cash allows users to deposit their cryptocurrency holdings in a privacy-preserving manner. The process begins with a user depositing a fixed amount of Ether (ETH) into the Tornado Cash smart contract. This deposit, denoted as N-ETH, is referred to as a "token." The critical aspect of this process is that the deposited amount can be later withdrawn with no direct link to the original transaction. Let's break down the key steps of a deposit:

- **Step 1: token Creation**

To initiate a deposit, a user generates two random numbers, k and r, each belonging to the set B248. These values are crucial as they form the basis for the token's privacy. The token's commitment, C, is calculated as $C = H_1(k||r)$, where H_1 is a Pedersen hash function. This commitment effectively conceals the token's true value and its ownership.

- **Step 2: Transaction Submission**

The user sends an Ethereum transaction with N ETH to the Tornado Cash smart contract, with C interpreted as an unsigned 256-bit integer. If the Merkle tree (T) is not full, the contract accepts the transaction and adds C as a new non-zero leaf. This ensures that the user's deposit becomes part of the pool of mixed funds.

5.1.2 The Withdrawal Process

The withdrawal process in Tornado Cash is designed to preserve user privacy while providing a mechanism for users to retrieve their funds. However, this process introduces a layer of complexity, particularly in situations where regulatory compliance and transaction tracing are necessary. Here's how a withdrawal works:

- **Step 1: User Initiation**

When a user wishes to withdraw their funds, they select a recipient address (A) and specify a fee value (f) where $f \leq N$. Additionally, they choose a root (R) among the stored ones in the contract and compute an opening $\big(O(l)\big)$ that ends with R. This root selection and opening computation are essential for the withdrawal process.

- **Step 2: Nullifier Calculation**

To ensure the privacy of the withdrawal, the user calculates a nullifier hash (h) as $H_1(k)$.

This nullifier hash is unique to the token being withdrawn and serves as a cryptographic proof that the user has the right to withdraw the funds.

- **Step 3: Proof Construction**

The user constructs a proof (P) using the deposit key pair (dp) and the relevant parameters, including the Merkle tree (T), k, r, the leaf index (l), A, f, and t (the Relayer address). This proof demonstrates their ownership of the token without revealing the token's commitment.

- **Step 4: Withdrawal Execution**

The withdrawal can be executed in one of two ways. The user can send an Ethereum transaction to the Tornado Cash contract, supplying R, h, A, f, t, and P in the transaction data. Alternatively, the user can send a request to a Relayer, providing the transaction data (R, h, A, f, t, P). The Relayer is then responsible for initiating the transaction with the contract.

5.1.3 Security and Privacy

Tornado Cash makes several security claims to ensure the privacy and integrity of the protocol:

- Only tokens deposited into the contract can be withdrawn.
- No token can be withdrawn twice.
- Any token can be withdrawn once if its parameters (k, r) are known unless a token with the same k has already been deposited and withdrawn.
- If k or r is unknown, a token can not be withdrawn.
- The proof is binding and can not be used with a different nullifier hash, recipient address, or fee amount.
- The cryptographic primitives used in Tornado Cash provide strong security guarantees.

While Tornado Cash's privacy solution is robust, it's essential to recognize that regulatory authorities may require transaction information for specific cases. This is where Night Protocol's integration of Multi-Party Computation (MPC) comes into play.

5.2 Privacy pools meet MPC

Vitalik Buterin (2023) introduced the concept of Privacy Pools, a scheme that allows users to prove the legitimacy of their transactions through associated sets. However, there are some issues with this approach. One major problem is that due to the difficulty of real-time analysis of the legitimacy of each transaction, the data of associated sets may be incomplete, causing transactions initiated by users to be excluded from the associated sets, leaving them unable to prove their innocence.

To address these issues, Night Protocol has adopted an improved solution based on MPC (Multi-Party Computation) and threshold decryption techniques to simultaneously protect user privacy and meet compliance and regulatory requirements. Here are the key steps of the improved solution:

1. Initialization of threshold keys based on MPC. This can involve multiple parties, such as DAO organization members or, in a simplified scenario, two parties consisting of the platform and regulatory authorities. This step ensures the security and decentralization of the keys.

2. Introduce a field called T in Deposit and Withdraw transactions for secret tracing. The field T includes the $secret$ information held by the user and a random number r. Field T is encrypted with threshold keys, and its content can only be decrypted by multiple parties. This ensures that sensitive information in transactions receives sufficient privacy protection.

3. When regulatory authorities need to trace a particular transaction (Deposit or Withdraw), they can run threshold decryption through MPC computations to decrypt the secret information of the transaction, revealing the destination of funds. This process provides the necessary compliance and regulatory transparency while safeguarding user privacy.

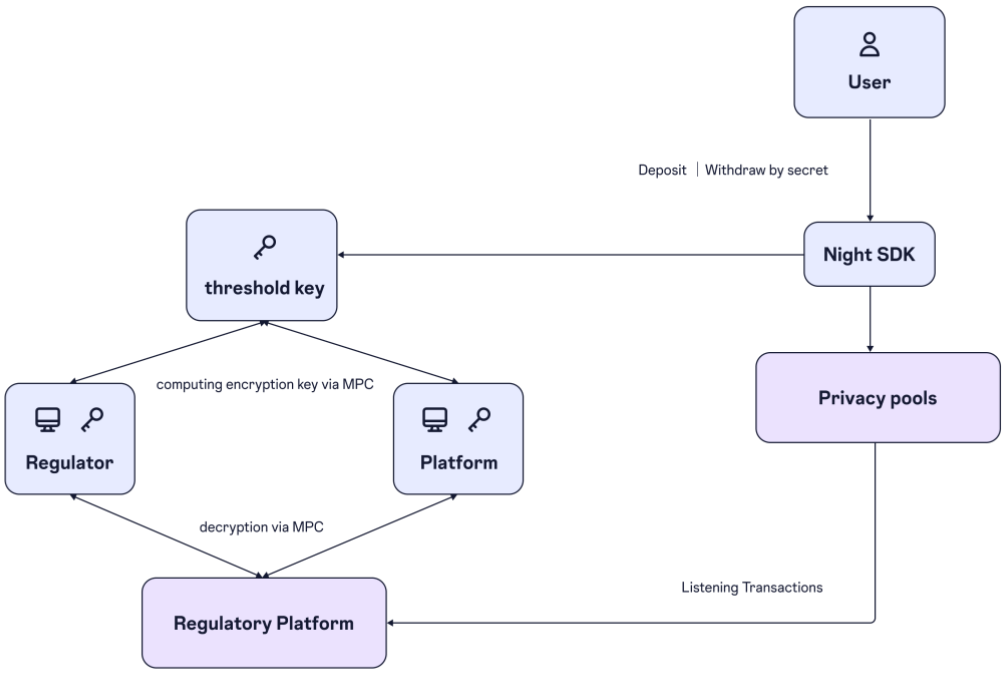


Figure 7. Regulatory platform based on MPC

The threshold algorithm based on MPC is described as follows：

The Night protocol implements a (2, 2) threshold cryptosystem (to be expanded to [m,n] in future iterations). Here, two-party EC-ElGamal scheme: Two-party computation of ciphertexts, the global decryption key is given by: $x = x\_1 + x\_2 \mod p$, in additive key share form. The global encryption key is given by $h = x * P$:

5.2.1 Notation

Symbol	Notion	Symbol	Notion
P	Elliptic curve base point	x	Global private key(no one knows it)(type: scalar)
p	Order of the base point	h	Global public key(type: ecpoint)
Z_n	Field of operations for elliptic curves	x_i	party-i 's private key (key share of)(type: scalar)
+	Addition operation in numerical terms	h_i	party-i 's public key (key share of)(type: ecpoint)
*	Multiplication operation in numerical terms	c_i	party-i 's commitment(type: scalar)
$\oplus$	Point addition operation on elliptic curves	r_i	Random number(type: scalar)
$\otimes$	Point doubling operation on elliptic curves	m	message
H	keccak256	ciphertext	ciphertext of m under AES with symmetric key
k_{point}	Point can derive the symmetric key	sym_key	symmetric key k

5.2.2 Phase 1: Global public key negotiation

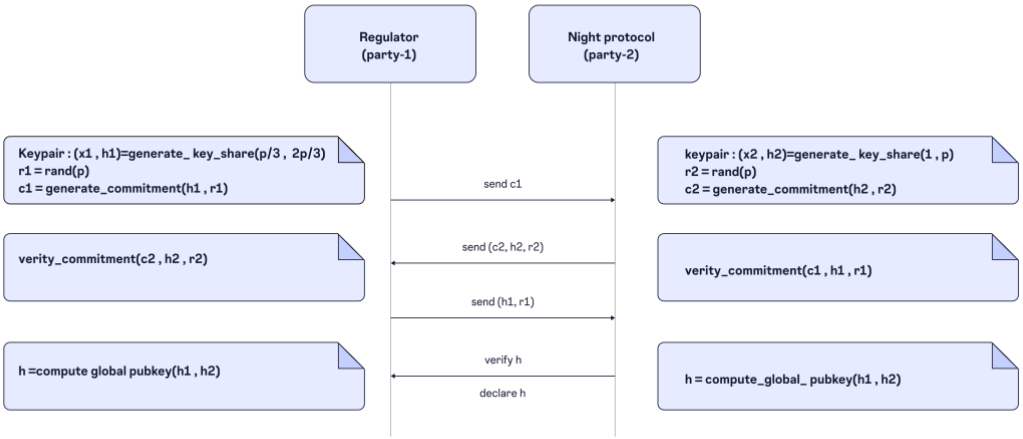
The threshold encryption public key negotiation goes through the following steps.

1. Generate the keypair (x_1, h_1) for party-1 regarding h and make a commitment $c_1 = H(h_1, r_1)$ for h_1. Generate keypair (x_2, h_2) for party-2 regarding h and make a commitment $c_2 = H(h_2, r_2)$ for $h2$.

Function	**Operation**
generate_key_share(m, n) at party-i	$x_i \xleftarrow{R} [m, n], h_i = x_i \otimes P$
rand(p) at party-i	$r \xleftarrow{R} [1, p]$
generate_commitment(m, n) at party-i	$c = H(m \,\|\| n)$
verify_commitment(c, m, n) at party-i	$c' = H(m \,\|\| n)$, check $c == c'$

2. Party-1 sends c_1 to party-2.
3. Party-2 sends c_2 and the preimage (h_2, r_2) of c_2 to party-1.
4. Party-1 verifies $c_2 = H(h_2, r_2)$ and then sends the preimage (h_1, r_1) of c_1 to party-2.
5. Party-2 verifies $c_1 = H(h_1, r_1)$.
6. Party-1 and party-2 each compute $h = h_1 + h_2$, confirm that the results are the same, and jointly announce the global encryption key as h .

Function	**Operation**
compute_global_pubkey(m,n) at party-i	$h = m \oplus n$



5.2.3 Phase 2: Encryption

The following process is standard hybrid encryption using EC-ElGamal, assuming that the encryption party has already obtained the global encryption key h through the following steps:

1. The encrypting party calls generate_sym_key(p) to generate a random k_{point}, and then calls compute_sym_key(k_{point}) to compute the symmetric key pair sym_key.

Function	**Operation**
generate_key_point(p) at party-i	$k \xleftarrow{R} [1, p], k_{point} = k \otimes P$
compute_sym_key(k_{point}) at party-i	$sym\_key = H\left(point2bytes\left(k_{point}\right)\right)$

2. The encrypting party calls the AES algorithm to encrypt the message m using the symmetric key sym_key to obtain the symmetric ciphertext enc, and then uses EC-ElGamal to encrypt by calling elgamal_encrypt(k_{point}, h) to obtain ($C1, C2$).

Function	**Operation**
elgamal_encrypt(k_{point}, h) at encrypt-party	$r \xleftarrow{R} [1, p], C_1 = r \otimes P, C_2 = k_{point} \oplus (r \otimes h)$

3. The ciphertext (ciphertext, $C1, C2$) is made public.

5.2.4 Phase 3: Threshold decryption

In case regulators initiate bad actor proceedings, the threshold cryptography protecting the raw data of the user can be recovered using the following steps:

1. Each party-i calculates the partial decryption D_i with respect to C_1.

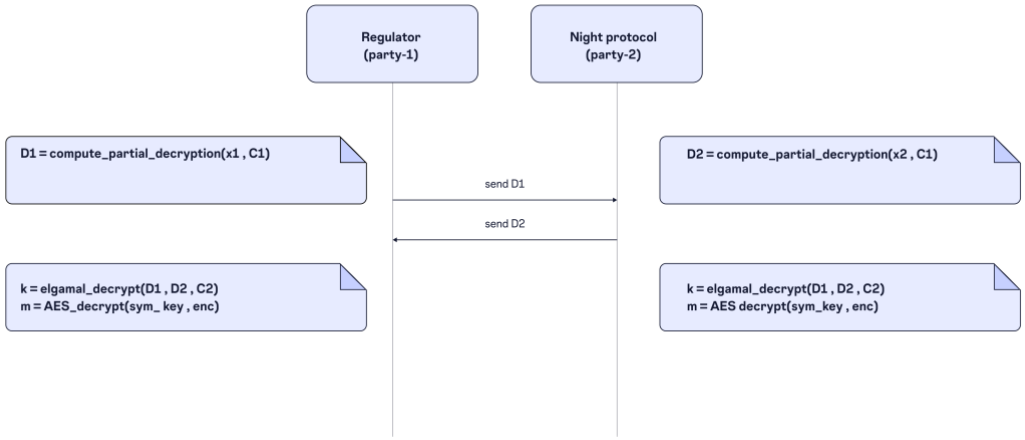
Function	Operation
compute_partial_decryption(xi, C1) at party-i	$D_i = x_i \otimes C_1$

2. Party-i sends D_i to party-3-i.

3. Party-i locally calls elgamal_decrypt(D1, D2, C2) to obtain , and then calls compute_sym_key(k_{point}) to compute the symmetric key pair sym_key.

Function	Operation
elgamal_decrypt(D1, D2, C2) at party-i	$D = D_1 \oplus D_2, k_{point} = C_2 \oplus (-D)$

4. Party-i calls the AES algorithm to decrypt the symmetric ciphertext enc using the symmetric key sym_key to obtain the message m.



The integration of Multi-Party Computation (MPC) solutions empowers the protocol to deliver uncompromising privacy protection for users while adroitly responding to the exigencies of regulatory compliance. This duality ensures the safeguarding of user rights and interests, paving the way for a future where blockchain technology can thrive within the bounds of legal and regulatory frameworks.

6. The Night Protocol Use Cases

The Night Protocol is an innovative solution designed to address the privacy concerns associated with transactions and interactions on the Ethereum blockchain. The protocol is built around the concept of stealth addresses, a cryptographic technique that enhances privacy without compromising the integrity of the blockchain.

At its core, the Night Protocol is composed of four main components:

Front-end Interface: This user-friendly interface enables users to conduct a variety of private transactions, including sending, receiving, swapping, staking, and lending tokens. The interface is designed to be intuitive and simple to use, even for those with limited experience with blockchain technology.

SDK (Stealth Address as a Service): The SDK provides a streamlined library that empowers dApp frontends to access user's private asset balances in Night-supporting wallets and transmit user operations for wallet authorization. This feature allows developers to easily integrate Night Protocol's privacy features into their own applications.

Night Snap: Night Snap is a widget that can be integrated into a user's MetaMask extension, facilitating interaction with Night-compatible applications. With Night Snap, users can take advantage of Night's privacy features without leaving their familiar MetaMask environment.

Private Smart Contract Wallet: The Night Protocol introduces a privacy-centric smart contract wallet. This wallet leverages account abstraction and stealth address technology to provide enhanced privacy to users. The wallet is also designed to offer seamless interaction with other Ethereum-based dApps and platforms.

6.1 Use Cases of Stealth Addresses

Stealth addresses have several practical use cases that leverage their enhanced privacy features. This privacy is particularly beneficial in situations where the anonymity of the sender, recipient, or both is crucial. Here are some of those use cases:

6.1.1 Private Payments

Stealth addresses allow for confidential payments to other stealth addresses. In this scenario, only the sender and recipient know about the transaction, maintaining privacy for both parties involved.

6.1.2 Payroll

Companies can use stealth addresses to handle their payroll. By doing this, they can maintain the confidentiality of their employees' wages and bonuses, ensuring that only the company and the individual employee know the specifics of the transaction.

6.1.3 Peer-to-Peer (P2P) Payments

Stealth addresses can be used for P2P payments where the anonymity of the sender and receiver is desired. This is especially useful in situations where users want to transact privately, away from the prying eyes of third parties.

6.1.4 Business Payments

Businesses can use stealth addresses to make private payments. This can be beneficial in a variety of situations, such as maintaining the confidentiality of business transactions, protecting supplier information, or safeguarding the details of financial deals.

6.2 Untraceable DApp Interactions

Stealth addresses enable users to engage with top Ethereum DApps seamlessly and privately. All transactions are not linked to the user's main wallet address, ensuring untraceability and enhancing user privacy. This can be especially beneficial for users of DeFi platforms, online games, and other DApps where privacy is a concern.

6.2.1 Private Swapping

Stealth addresses can be used for private swapping of tokens. This allows users to swap tokens without revealing their main wallet addresses, increasing privacy and security.

6.2.2 Private Lending

Stealth addresses can also be used in the context of private lending. Borrowers and lenders can transact privately, ensuring that the details of their financial transactions remain confidential.

6.2.3 Private Staking

Finally, stealth addresses can be used for private staking. In this context, users can stake their tokens in a private and secure manner, ensuring that their staking activities cannot be traced back to their main wallet addresses.

7. Security Implications

In this section, we delve into the intricate security implications that underpin the Night Protocol's core functionality. Our focus centers on two critical aspects: quantum-resistant security and protection against Denial-of-Service (DoS) attacks. These considerations are instrumental in ensuring the protocol's robustness and resilience in the face of potential threats. The discussion is largely based on the insights provided by Vitalik Buterin.

7.1 Quantum-Resistant Security and Stealth Addresses

The scheme explained above leverages the power of elliptic curves, which, while incredibly effective, are unfortunately vulnerable to quantum computers. If quantum computers become a significant threat, we would need to switch to quantum-resistant algorithms. There are two natural candidates for this: elliptic curve isogenies and lattices.

7.1.1 Elliptic Curve Isogenies

Elliptic curve isogenies are a very different mathematical construction based on elliptic curves. They possess linearity properties that allow us to perform similar cryptographic tricks to those we've discussed above but cleverly avoid creating cyclic groups that might be vulnerable to discrete logarithm attacks with quantum computers.

The primary weakness of isogeny-based cryptography is its highly complicated underlying mathematics and the risk that possible attacks are hidden under this complexity. Some isogeny-based protocols were broken last year, although others remain safe. The main strength of isogenies lies in the relatively small key sizes and the ability to directly port over many types of elliptic curve-based approaches.

7.1.2 Lattices

Lattices represent a very different cryptographic construction that relies on far simpler mathematics than elliptic curve isogenies and are capable of some very powerful techniques (e.g., fully homomorphic encryption). Stealth address schemes could be built on lattices, though designing the best one is an open problem. However, lattice-based constructions tend to have much larger key sizes.

7.2 Protection Against DoS Attacks

Ensuring that the parsing process remains efficient and resource-friendly is pivotal for Night Protocol's success. However, the protocol's reliance on off-chain execution of several elliptic curve (EC) operations, which bypasses blockchain-associated gas costs, renders it susceptible to DoS attacks. Malicious actors can inundate the network with a barrage of announcements, forcing users to perform redundant EC operations on false announcements, needlessly consuming computational resources. Furthermore, the costs incurred by sending an announcement may be lower than the costs associated with parsing it, introducing inefficiencies that could adversely affect the user experience by needlessly prolonging the parsing process.

To address this vulnerability, Night Protocol explores two distinct approaches for mitigating DoS attacks based on Wahrstatter et al. (2023) proposed solutions: the toll-based approach and the staking-based approach. In the following sections, we elaborate on both methodologies and highlight why the staking-based approach aligns better with the protocol's objectives.

7.2.1 Toll-Based Approach

The toll-based approach focuses on introducing a financial deterrent to mitigate the risk of DoS attacks. This deterrent takes the form of a "toll" (T) that accounts for the computational costs incurred during the parsing process. Specifically, the toll covers parsing costs, including EC multiplications and hashing operations. The sender, who generates announcements, attaches the toll to the transaction and bears the associated costs. This strategy ensures that the parsing costs related to announcements are shared between senders and recipients, preventing a disproportionate burden on recipients. The toll serves as a financial barrier, discouraging potential attackers from flooding the network with spam announcements.

The specific value of the toll is a critical consideration, taking into account various factors such as network conditions, the overall cost structure of the protocol, and the desired level of protection against spamming. While the primary purpose of the toll is to deter spamming economically, it is not necessary to cover the entire parsing process's costs. Therefore, the toll can be significantly reduced while still effectively deterring DoS attacks. To maintain the integrity of DoS attack prevention, it is crucial to ensure that the collected toll does not flow directly back to the originator.

One proposed option is to direct the collected toll to the coinbase address of the respective block, typically belonging to the block proposer. This approach would distribute the toll among block proposers, providing them with additional incentives to include stealth address transactions in blocks. Consequently, block proposers would have an added source of extractable value, incentivizing them to prioritize stealth address transactions during block creation. However, it is essential to note that initiators of stealth address transactions may offset the toll expenses by reducing the gas price, potentially undermining the effectiveness of toll-based spam prevention.

To determine the optimal toll value and its utilization in different network environments and circumstances, further research and analysis are necessary. It is also critical to ensure that the toll mechanism does not introduce trust requirements or centralizing vectors.

7.2.2 Staking-Based Approach

The staking-based approach in Night Protocol capitalizes on the protocol's dual-key setup, allowing users to share private scanning keys with third-party entities specializing in parsing. These third-party providers can offer parsing services at a market price and implement defense measures against DoS attacks. These measures may be based on specific heuristics designed to identify potential spammers effectively. As a result, third-party parsing providers contribute additional layers of protection against DoS attacks targeting users, bolstering the reliability and effectiveness of the parsing process.

A staking system can be introduced to empower parsing providers with a supplementary tool for managing spam and countering Sybil attacks. In this system, users have the option

to stake a predetermined amount of ETH and lock it within the contract, often referred to as the Announcer contract. Parsing providers can subsequently verify whether the sender of a stealth address transaction has staked the requisite collateral. If not, parsing providers have the flexibility to deprioritize the announcements from that sender when serving them to parsing users.

This staking system can be defined as follows:

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

This staking-based approach leverages two priority factors: $PF1$, based on the amount of ETH staked, and PF2, based on the number of announcements made by a user.

1. $PF1$: Staking Priority Factor

For each user $u_i \in U$ and their corresponding deposited ETH amount $D(u_i)$, the staking priority factor ($PF1$) is defined 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 equal to the MIN_STAKE_VALUE.

1. $PF2$: 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 ($PF2$) is determined as:

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

To discourage spamming, a higher announcement count priority factor ($PF2$) is assigned to users who have made fewer announcements:

$$PF2(u_i) = 1/n(u_i)$$

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

To obtain a comprehensive prioritization factor (PF) for each user, we combine $PF1$ and $PF2$, adjusting for their relative importance using weights w_1 and w_2:

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

The initial values of w_1 and w_2 are both set to 1, providing equal weight to ETH staked and the number of announcements made. However, these weights can be adjusted to emphasize one factor over the other based on specific prioritization needs.

Parsing providers can then use the computed PF values to order the list of announcements, ensuring that announcements from staking users receive higher priority in the parsing process. Additionally, announcements from users with fewer previous announcements are given precedence.

One significant advantage of the staking-based DoS attack prevention is that it is implemented on the parsing side, allowing parsing providers to manage spam effectively. This approach enables parsing providers to disregard or deprioritize announcements from spamming users when serving them to parsing users. Furthermore, by imposing a stake requirement for prioritization, Sybil attacks become inefficient, and the stakes of known spammers can be traced, discouraging them from switching addresses to evade deprioritization.

Another notable benefit of the staking approach is that it does not impose any costs on users. The minimum required stake can be locked directly in the contract communicating with the Announcer contract within the transaction that interacts with a stealth address, ensuring a seamless user experience.

In light of the considerations discussed, we assert that the staking-based approach aligns better with the specific goals and requirements of Night Protocol compared to the toll-based approach, which would entail a toll for every stealth address transaction.

8. Future Improvements and Features for the Night Protocol

The Night Protocol, as a stealth address solution, has the potential for numerous future improvements and feature additions, especially as it pertains to the Ethereum ecosystem.

8.1 Quantum-Resistant Security

As discussed earlier, the advent of quantum computing poses a significant threat to the current cryptographic systems, including elliptic curve cryptography used by the Night Protocol. Developing quantum-resistant security measures, such as those based on elliptic curve isogenies or lattices, will be a crucial step in ensuring the long-term viability of the Night Protocol.

8.2 Efficient Use of STARKs

The use of STARKs (Scalable Transparent ARguments of Knowledge) in the Night Protocol presents both opportunities and challenges. While STARKs can provide robust security and privacy features, they are also resource-intensive. Future improvements can focus on developing more efficient ways to use STARKs, such as the proposed aggregation protocol to combine multiple STARKs into a single recursive STARK.

8.3 Enhanced Privacy Features

While stealth addresses inherently provide significant privacy benefits, there is always room for enhancements. Future improvements could focus on developing additional privacy features, such as more sophisticated obfuscation techniques or enhanced transaction mixing capabilities.

8.4 Integration with Other Protocols and DApps

The Night Protocol could also potentially integrate with other protocols and DApps in the Ethereum ecosystem. This would allow users to take advantage of the privacy features of the Night Protocol across a wider range of applications and services.

8.5 Final Thoughts on the Potential Impact of the Night Protocol

The Night Protocol holds significant promise for the Ethereum ecosystem. By providing a reliable and robust mechanism for generating and using stealth addresses, the protocol can greatly enhance user privacy.

The potential applications of the Night Protocol are wide-ranging. From private payments and confidential business transactions to seamless and private interactions with DApps, the protocol can help make the Ethereum ecosystem more secure and user-friendly.

Moreover, the Night Protocol could play a crucial role in advancing the state of privacy technology in the blockchain space. By pushing the boundaries of what's possible with stealth addresses and privacy-preserving technologies, it could inspire further innovations in this area.

However, as with any new technology, the Night Protocol will also face challenges, particularly in terms of ensuring long-term security against emerging threats like quantum computing. It will be interesting to see how the protocol evolves to meet these challenges and realize its full potential in the years to come.

References

Albrecht, M. R., Grassi, L., Rechberger, C., et al. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity." ASIACRYPT (1). Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 191–219.

AbdulKader, M. M., & Kumar, S. G. "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.

ByteCoin. "Untraceable Transactions Which Can Contain a Secure Message Are Inevitable." 2011 Bitcoin Forum. https://bitcointalk.org/index.php?topic=5965.0.

Buterin, Vitalik. "An Incomplete Guide to Stealth Addresses." 2023. https://vitalik.ca/general/2023/01/20/stealth.html.

Buterin, Vitalik. "The Three Transitions." 2023. https://vitalik.ca/general/2023/06/09/three_transitions.html.

Buterin, Vitalik, et al. "Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium." September 6, 2023. Available at SSRN or http://dx.doi.org/10.2139/ssrn.4563364.

Buterin, Vitalik, et al. "EIP 4337: Account Abstraction Using Alt Mempool." https://eips.ethereum.org/EIPS/eip-4337, September 2021.

Courtois, N. T., & Mercer, R. "Stealth Address and Key Management Techniques in Blockchain Systems." In Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP (pp. 559–566). INSTICC. SciTePress, 2017.

Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). "Zerocash: Decentralized anonymous payments from bitcoin." In 2014 IEEE Symposium on Security and Privacy (pp. 459–474). IEEE.

Fan, J., Wang, Z., Luo, Y., Bai, J., Li, Y., & Hao, Y. "A New Stealth Address Scheme for Blockchain." Proceedings of the ACM Turing Celebration Conference - China, ACM TURC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online] https://doi.org/10.1145/3321408.3321573.

Fan, X. "Faster Dual-Key Stealth Address for Blockchain-Based Internet of Things Systems." In Blockchain – ICBC 2018 (pp. 127–138). Springer International Publishing, 2018.

Feng, C., Tan, L., Xiao, H., Yu, K., Qi, X., Wen, Z., & Jiang, Y. "Pdksap: Perfected Double-Key Stealth Address Protocol Without Temporary Key Leakage in Blockchain." 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops), 2020, pp. 151–155.

Feng, C., Tan, L., Xiao, H., Qi, X., Wen, Z., & Liu, Y. "Edksap: Efficient Double-Key Stealth Address Protocol in Blockchain." 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2021, pp. 1196–1201.

Liu, Z., Nguyen, K., Yang, G., Wang, H., & Wong, D. S. "A Lattice-Based Linkable Ring Signature Supporting Stealth Addresses." In Computer Security – ESORICS 2019 (pp. 726–746). Cham: Springer International Publishing, 2019.

"The Monero Project." 2014 https://www.getmonero.org/.

Pertsev, A., Semenov, R., and Storm, R. "Tornado Cash Privacy Solution Version 1.4." 2019.

Saberhagen, N. "Cryptonote v 2.0." Oct 2013. https://web.archive.org/web/20201028121818/https://cryptonote.org/whitepaper.pdf.

Wahrstätter, Anton, et al. "BaseSAP: Modular Stealth Address Protocol for Programmable Blockchains." arXiv preprint arXiv:2306.14272 (2023).