Stealth Addresses on EVM Blockchains: A Cryptographic, Protocol, and Application-Layer Analysis

Section 1: The Imperative for On-Chain Privacy on Public Ledgers

1.1 Deconstructing Pseudonymity: The Limits of Address-Based Privacy

Public blockchains, such as Ethereum and other EVM-compatible networks, are often misconstrued as anonymous systems. In reality, they operate on a principle of pseudonymity, a distinction that is fundamental to understanding the need for advanced privacy solutions like stealth addresses.[1] The entire state of the blockchain is a public, distributed ledger, meaning that every transaction—including the sender's address, the recipient's address, the transaction amount, and the specific assets involved—is openly visible to any participant or observer.[3]

An address, such as 0xAb5801a7D398351b8bE11C439e05C5B3259aeC9B, functions as a pseudonym. It does not inherently contain personally identifiable information (PII).[4] However, this veil of privacy is exceptionally thin. The moment an address is linked to a real-world identity, its entire history of interactions—both past and future—becomes deanonymized.[5] This linkage can occur through numerous common activities: withdrawing funds from a Know Your Customer (KYC) compliant centralized exchange, associating an address with a public profile or an Ethereum Name Service (ENS) domain like

vitalik.eth, or simply by publicly posting an address to receive donations.[1] Once this link is established, the pseudonym is broken, and the address becomes a transparent record of

financial activity.

1.2 The Threat of Transaction Graph Analysis and Deanonymization

The transparency of public ledgers has given rise to a sophisticated industry of blockchain analytics. These services and actors employ transaction graph analysis to map the flow of funds across the network, identifying patterns and linking addresses that interact with each other.[7] By applying heuristics such as the common-input-ownership heuristic (which assumes that multiple input addresses in a single transaction belong to the same entity), analysts can cluster addresses and build comprehensive profiles of a user's financial behavior, even without an initial PII link.[7]

This level of traceability poses significant risks that extend beyond mere financial privacy.[8] For individuals, it can lead to targeted phishing attacks, social engineering, or even physical threats if large holdings are identified.[8] For businesses, it creates a landscape of radical, involuntary transparency. Competitors could analyze payroll transactions to deduce salary structures and poach key employees, track payments to suppliers to reverse-engineer supply chains, or monitor treasury management strategies.[10] In politically sensitive contexts, it could expose donors to activist groups or journalists to repressive regimes, creating a chilling effect on free association and expression.[2]

The maturation of these on-chain surveillance capabilities has created a "privacy deficit." The initial, naive perception of blockchain as an anonymous system has been replaced by the reality that it can be even more transparent than the traditional banking system. This has driven a clear market need for cryptographic solutions that can restore meaningful privacy guarantees to users. Stealth addresses are a direct and powerful response to this deficit, aiming to re-establish the confidentiality expected in financial interactions.

1.3 Establishing the Need for Recipient-Side Privacy Solutions

The landscape of blockchain privacy technologies is diverse, with different solutions targeting different aspects of a transaction. For instance, technologies like ring signatures, famously used by Monero, are designed to obscure the *sender* of a transaction by mixing their cryptographic signature with a set of decoys.[5] Other systems, like those based on zero-knowledge proofs, can hide the sender, receiver, and transaction amount simultaneously.

Stealth addresses, however, are specifically engineered to solve the problem of *recipient anonymity*.[13] They operate on the principle that while the sender's action of sending funds is public, the identity of the ultimate recipient should be concealed. This provides a unique set of privacy guarantees tailored to specific use cases where the recipient's identity might be public, but their financial activity must remain private. A prime example is a public charity, a political campaign, or an open-source project seeking donations. The organization can publicly share a single, stable identifier (its stealth meta-address), but each donation will be sent to a unique, unlinked on-chain address. This prevents observers from tracking the total amount of funds raised, monitoring how those funds are subsequently spent, or analyzing the donation patterns of supporters.[5] In essence, stealth addresses break the linkability of incoming payments, protecting the recipient's financial sovereignty.

Section 2: Foundational Principles of Stealth Address Technology

2.1 Conceptual Framework: One-Time Addresses for Unlinkable Receipts

At its core, a stealth address is a unique, one-time-use address generated for every single transaction.[1] It is derived cryptographically by the sender on behalf of the recipient in a non-interactive manner. Although this new address appears random and unassociated with any known entity on the blockchain, it is exclusively controllable by the intended recipient.[13] This mechanism effectively severs the public on-chain link between the recipient's main identity and the transaction itself.

This can be conceptualized as a digital equivalent of a post office box or a proxy address.[3] Instead of giving out a permanent home address (the main wallet), the recipient provides a method for others to generate a unique, single-use mailing address for each package. All packages arrive at their distinct destinations, but only the recipient has the master key to know which boxes belong to them and to unlock them. This prevents an observer from determining how many packages the recipient has received or from whom by simply watching the post office. Similarly, stealth addresses ensure that multiple payments to the same recipient are directed to different on-chain addresses, making it computationally infeasible for an external party to link these transactions together or to the recipient's primary wallet.[3]

2.2 Historical Context: From Bitcoin Proposals to Monero's Implementation

The concept of stealth addresses is not new to the EVM ecosystem. It was first formally proposed for the Bitcoin network by Peter Todd in 2014 as a solution to the privacy erosion caused by address reuse.[3] While adoption in the Bitcoin ecosystem has been limited, the technology found its most robust and widespread implementation as a core component of the privacy-centric cryptocurrency Monero (XMR).[2]

In Monero, stealth addresses are not an optional feature but a mandatory part of the protocol, ensuring privacy by default for all users.[12] They work in concert with two other key privacy technologies: Ring Signatures, which obscure the sender's identity, and Ring Confidential Transactions (RingCT), which hide the transaction amount.[5] This three-tiered system provides comprehensive privacy, and Monero's successful, long-standing implementation serves as a powerful proof-of-concept for the viability and security of the underlying cryptographic principles that are now being standardized for EVM chains.

2.3 The Dual-Key Model: Separating Spending and Viewing Capabilities

A critical architectural innovation that makes stealth addresses practical and secure is the dual-key model.[12] Instead of a single private key controlling all aspects of an account, as is typical with a standard Externally Owned Account (EOA), a stealth address user generates two distinct keypairs:

1. **Spending Keypair (kspend, Pspend):** The private spending key, kspend, is the ultimate source of authority. It is mathematically required to derive the final private key for each stealth address and is used to sign transactions to move funds *out* of those addresses. Due to its critical importance, the spending key should be handled with the highest level of security, ideally kept in cold storage or a hardware wallet and never exposed to an online environment.[15] The public spending key,
   Pspend, is part of the publicly shared meta-address.
2. **Viewing Keypair (kview, Pview):** The private viewing key, kview, has a single, specific purpose: to scan the blockchain and *detect* incoming transactions. It is used to compute the shared secret with the sender's ephemeral public key, but it cannot be used to derive the final spending keys or authorize any transactions.[12] The public viewing key,

Pview, is the other component of the meta-address.

This separation of concerns is not merely a feature but a fundamental prerequisite for the system's usability and security. The primary operational burden for a stealth address recipient is the need to constantly scan the blockchain for relevant transactions.[17] A naive, single-key implementation would require the master private key to be active and online to perform this scanning, creating an unacceptable security risk; a compromise of this key would lead to the loss of all current and future funds.

The dual-key model elegantly solves this problem by creating a form of role-based access control for the user's own funds. It separates the "read" capability (viewing key) from the "write" capability (spending key). A user can safely load their private viewing key into a "hot" wallet, a browser extension, or even delegate the scanning process to a trusted third-party service without exposing the private spending key.[15] This allows for efficient discovery of incoming funds while the keys that control those funds remain securely offline, making the entire system practical for real-world use.

Section 3: A Cryptographic Deep Dive: The Mechanics of Stealth Address Generation

The security and privacy guarantees of stealth addresses are rooted in the mathematical properties of Elliptic Curve Cryptography (ECC), the same cryptographic foundation that secures all standard Ethereum accounts. The process leverages a non-interactive form of the Elliptic Curve Diffie-Hellman (ECDH) key exchange to allow a sender and receiver to derive a shared secret, which is then used to generate a unique, one-time address and its corresponding private key.

3.1 Primer on Elliptic Curve Cryptography (ECC) over Secp256k1

Ethereum, like Bitcoin, utilizes the secp256k1 elliptic curve for its public-key cryptography.[21] An elliptic curve is defined by an equation over a finite field, and its points exhibit a group structure under an operation called "point addition." The core operation relevant to key generation is scalar multiplication.

- **Generator Point (G):** A publicly known, standardized point on the secp256k1 curve.[21]
- **Private Key (k):** A large, randomly generated integer. This is the user's secret.

- **Public Key (P):** The result of performing scalar multiplication of the generator point G by the private key k. This is expressed as $P=k \cdot G$. The public key is a point on the curve, represented by its (x, y) coordinates.[21]

The fundamental security assumption of ECC is the Elliptic Curve Discrete Logarithm Problem (ECDLP). It states that while it is computationally trivial to calculate the public key P given the private key k and the generator G, it is computationally infeasible to determine the private key k given only the public key P and the generator G.

3.2 The Elliptic Curve Diffie-Hellman (ECDH) Key Exchange Protocol

The ECDH protocol leverages a key property of scalar multiplication: it is commutative. This allows two parties to establish a shared secret over an insecure channel without exchanging their private keys.[3]

Let's say Alice has a private key kA and public key $PA=kA \cdot G$, and Bob has a private key kB and public key $PB=kB \cdot G$. They exchange their public keys.

- Alice computes a shared secret point S by multiplying her private key with Bob's public key: $S=kA \cdot PB=kA \cdot (kB \cdot G)$.
- Bob computes a shared secret point S by multiplying his private key with Alice's public key: $S=kB \cdot PA=kB \cdot (kA \cdot G)$.

Due to the associative property of scalar multiplication, both Alice and Bob arrive at the exact same point $S=(kA \cdot kB) \cdot G$, which is their shared secret.[24] An eavesdropper who only knows

PA and PB cannot compute S without solving the ECDLP.

3.3 Generating the Shared Secret: The Non-Interactive Workflow

Stealth addresses adapt the ECDH protocol for a non-interactive setting, where the sender (Alice) generates the shared secret without any real-time communication with the recipient (Bob).

1. **Recipient's Setup (Bob):** Bob generates two private keys: a private spending key kspend and a private viewing key kview. He computes their corresponding public keys, $Pspend=kspend \cdot G$ and $Pview=kview \cdot G$. The pair of public keys, (Pspend,Pview), constitutes his **stealth meta-address**, which he makes publicly available.[17]

2. **Sender's Action (Alice):** To send a payment to Bob, Alice performs the following steps:
    - She generates a new, single-use random private key known as an **ephemeral key**, denoted as r. This key is used only for this one transaction and is then discarded.[13]
    - She computes the corresponding **ephemeral public key**, R=r·G. This public key will be published on-chain as part of the transaction's announcement data.
3. **Shared Secret Calculation:**
    - Alice (Sender): Alice computes the shared secret point S by multiplying her ephemeral private key r with Bob's public viewing key Pview:

      S=r·Pview
    - Bob (Recipient): At a later time, when Bob is scanning the blockchain, he will see the ephemeral public key R that Alice published. He can then independently compute the exact same shared secret point S by multiplying his private viewing key kview with Alice's ephemeral public key R:

      S=kview·R

This works because r·Pview=r·(kview·G)=kview·(r·G)=kview·R. This elegant cryptographic trick is the foundation of the entire system, allowing for the asynchronous and non-interactive creation of a secret known only to the sender and recipient.[17]

3.4 Key Derivation: From Shared Secret to Stealth Address Private Key

The shared secret S is a point on the elliptic curve. To be used in the final key calculation, it must be converted into a scalar (a simple number). This is achieved using a Key Derivation Function (KDF), which in this context is simply a cryptographic hash function like Keccak-256.[26]

1. Hashing the Secret: Both Alice and Bob hash the shared secret point S to produce a shared secret scalar, sh:

   sh=hash(S)

   Since both parties computed the same point S, they will also compute the same scalar sh.17
2. Deriving the Stealth Public Key and Address (Alice's side): Alice can now compute the public key of the one-time stealth address. She takes Bob's public spending key Pspend and adds the elliptic curve point derived from the shared secret scalar sh:

   Pstealth=Pspend+(sh·G)

The final Ethereum stealth address, Astealth, is then derived from this public key using the standard Ethereum method (the last 20 bytes of the Keccak-256 hash of the public key).17

3. Deriving the Stealth Private Key (Bob's side): Bob is the only person who can compute the private key corresponding to Pstealth. He does this by taking his private spending key kspend and adding the shared secret scalar sh to it:

   kstealth=(kspend+sh)(modn)

   (where n is the order of the curve)

The correctness of this scheme is demonstrated by showing that the derived private key corresponds to the derived public key:

kstealth·G=(kspend+sh)·G=(kspend·G)+(sh·G)=Pspend+(sh·G)=Pstealth

This process ensures that Alice can generate a valid Ethereum address to which she can send funds, while only Bob, the holder of kspend, can compute the private key needed to control those funds.17

Section 4: Standardization on EVM Chains: An Analysis of ERC-5564 and ERC-6538

For stealth addresses to be a viable and interoperable privacy solution on EVM chains, a standardized protocol is essential. Without a common framework, each wallet and dApp would implement its own cryptographic variations and on-chain communication methods, creating a fragmented and unusable ecosystem.[15] Two Ethereum Improvement Proposals (EIPs), now Ethereum Request for Comments (ERCs), provide this crucial standardization: ERC-5564 defines the core protocol for announcing stealth transactions, and ERC-6538 defines a registry for discovering users' stealth meta-addresses.

4.2 ERC-5564: The Stealth Address Messenger Protocol

ERC-5564 establishes a minimal, universal on-chain footprint for stealth address transactions. It is designed to be crypto-system-agnostic, meaning it can support various underlying

cryptographic schemes (e.g., secp256k1, lattice-based crypto), but it provides a reference implementation for the secp256k1 curve used by Ethereum.[15]

The ERC5564Announcer Singleton Contract

The centerpiece of ERC-5564 is the ERC5564Announcer, a singleton smart contract deployed at the same canonical address (0x55649E01B5Df198D18D95b5cc5051630cfD45564) across all EVM-compatible chains.[29] This contract serves as a standardized, public bulletin board. Its sole purpose is to emit an

Announcement event whenever a stealth transaction occurs, creating a single, predictable on-chain location for all recipients to monitor.[25]

Dissecting the Announcement Event

The on-chain communication relies entirely on a single event, Announcement. A recipient's wallet listens for this event to discover incoming payments. Its parameters are:

- uint256 indexed schemeId: An identifier for the cryptographic scheme being used (e.g., 1 for the default secp256k1 with view tags).
- address indexed stealthAddress: The one-time stealth address that received the assets.
- address indexed caller: The address that initiated the announce call.
- bytes ephemeralPubKey: The sender's ephemeral public key (R). This is the critical piece of data the recipient needs to compute the shared secret.
- bytes metadata: An arbitrary data field used to convey additional information, most importantly the view tag.[31]

Metadata Structure and the View Tag

The metadata field is not arbitrary in practice; it has a defined structure to optimize the scanning process. Its first byte is reserved for the **view tag**. As described in the cryptographic section, the shared secret S is hashed to a scalar sh. The view tag is simply the first byte of this hash: v=sh.[32]

When a recipient's wallet scans an Announcement event, it first performs the cheapest possible check: it computes its own version of the view tag using its private viewing key and the event's ephemeralPubKey. If this locally computed view tag does not match the one in the metadata, the wallet knows with high probability (255/256) that this transaction is not for them and can discard it without performing the much more computationally expensive elliptic curve operations.[25] This optimization is crucial for making client-side scanning feasible. The remaining bytes of the

metadata field can be used to encode information about the asset transfer, such as the token contract address and the amount or token ID, which helps the recipient's wallet display the transaction details without having to query the stealth address's balance separately.[25]

Core Functions (Conceptual Interface)

While the primary on-chain interaction is the announce function that emits the event, ERC-5564 defines a conceptual interface that all compliant libraries and SDKs must implement off-chain. These functions encapsulate the cryptographic logic from Section 3:

- generateStealthAddress(stealthMetaAddress): Takes the recipient's meta-address and returns the generated stealthAddress, ephemeralPubKey, and viewTag.
- checkStealthAddress(stealthAddress, ephemeralPubKey, viewingKey, spendingPubKey): Takes the data from an Announcement and the recipient's keys to verify if the transaction belongs to them.
- computeStealthKey(stealthAddress, ephemeralPubKey, viewingKey, spendingKey): Takes the same inputs and computes the final private key for the stealth address, to be used for spending.[32]

4.3 ERC-6538: The Stealth Meta-Address Registry

While ERC-5564 provides the mechanism for announcing a stealth transaction, it doesn't solve the problem of how a sender discovers a recipient's stealth meta-address in the first place. ERC-6538 addresses this by defining a standard for an on-chain, public registry.[35]

Linking Identities to Keys

ERC-6538 specifies the ERC6538Registry, another singleton contract that acts as a public key directory. It allows users to create a public, on-chain link between their existing identifier (such as their EOA or ENS name) and their stealth meta-address (the concatenated public spending and viewing keys).[15]

Contract Interface and Functions

The registry contract provides a simple yet powerful interface:

- stealthMetaAddressOf(address registrant, uint256 schemeId): A public mapping that allows anyone to look up the registered stealth meta-address for a given user and cryptographic scheme.
- registerKeys(uint256 schemeId, bytes calldata stealthMetaAddress): A function that allows a user to register their own stealth meta-address.
- registerKeysOnBehalf(...): A function that allows a third party to register a meta-address for a user, provided they supply a valid EIP-712 signature from that user, enabling gasless setup flows.[35]

The User Workflow

Together, these two standards create a fluid user experience. A sender, Alice, no longer needs to ask Bob for a long, complex meta-address string. Instead, she can simply enter bob.eth into her wallet. The wallet application would then perform the following steps in the background:

1. Resolve bob.eth to Bob's Ethereum address via the ENS protocol.
2. Query the stealthMetaAddressOf mapping on the ERC6538Registry contract with Bob's address to fetch his stealth meta-address.
3. Proceed with the ERC-5564 workflow to generate the stealth address and create the transaction and announcement.[17]

This two-part standardization separates the concerns of discovery (ERC-6538) and announcement (ERC-5564), creating a modular and extensible framework for private transactions on the EVM.

Table: Key Components of the ERC-5564/ERC-6538 Framework

Component	Governing Standard	Purpose	Data Content / On-Chain Representation
Stealth Meta-Address	ERC-6538	Recipient's public identity used by senders for key derivation.	bytes string, typically concat(P_spend, P_view).
Ephemeral Public Key	ERC-5564	Sender's one-time public key, published for the recipient to derive the shared secret.	bytes string representing an ECC point, R.
ERC6538Registry	ERC-6538	An on-chain, singleton contract that maps user identifiers (addresses) to their stealth meta-addresses.	Contract at 0x6538... with stealthMetaAddressOf mapping.
ERC5564Announcer	ERC-5564	An on-chain, singleton contract that serves as a public bulletin board for stealth transactions by emitting events.	Contract at 0x5564... with announce function.
Announcement Event	ERC-5564	The standardized log entry that contains all necessary data for a recipient to detect a payment.	event Announcement(schemeId, stealthAddress, caller, ephemeralPubKey,

			metadata)
View Tag	ERC-5564	A 1-byte cryptographic hint to allow for rapid, computationally cheap filtering of irrelevant announcements.	The first byte of the metadata field in the Announcement event; hash(S).

Section 5: The End-to-End Transaction Lifecycle

Understanding the complete lifecycle of a stealth address transaction—from the recipient's initial setup to the sender's payment and the recipient's final recovery of funds—is crucial for appreciating both its elegance and its practical complexities.

5.1 Recipient Setup: A One-Time Process

For a user (Bob) to begin receiving stealth payments, their wallet must perform a one-time setup process. This establishes the foundational keys that will be used for all future incoming transactions.

1. **Step 1: Key Generation.** The wallet application first needs to generate the two core private keys: the spending key (kspend) and the viewing key (kview). To ensure these keys are recoverable and tied to the user's main identity, they are typically derived deterministically from the user's master seed phrase (the same one used for their regular EOA) by signing a predefined, constant string. This signature is then used as entropy to generate the two private keys, ensuring that the same keys can be regenerated if the user restores their wallet on a new device.[15]
2. **Step 2: Meta-Address Construction.** From the private keys kspend and kview, the wallet computes the corresponding public keys, Pspend and Pview, via scalar multiplication. The stealth meta-address is then constructed, typically by compressing and concatenating these two public keys.[15]
3. **Step 3: On-Chain Registration (Optional but Recommended).** For discoverability, the wallet should then prompt Bob to register this newly created meta-address on the public

ERC6538Registry. This involves a single on-chain transaction that calls the registerKeys function, creating a permanent link between Bob's primary address (e.g., bob.eth) and his stealth meta-address. This step makes it possible for others to send him stealth payments without needing to ask him for the long meta-address string directly.[18]

5.2 The Sender's Workflow: Non-Interactive Payment

The sender's (Alice's) experience is designed to be as seamless and non-interactive as possible, abstracting away the underlying cryptographic complexity.[15]

1. **Step 1: Recipient Discovery.** Alice opens her stealth-enabled wallet or dApp and inputs Bob's public identifier, such as bob.eth. The application automatically queries the ERC6538Registry to retrieve Bob's registered stealth meta-address.[18] If Bob hasn't registered, Alice would need to obtain the meta-address from him through an off-chain channel.
2. **Step 2: Stealth Address Generation.** Alice's wallet performs all the necessary cryptographic operations locally and instantaneously. It generates a random ephemeral private key (r), computes the ephemeral public key (R), calculates the shared secret with Bob's public viewing key, hashes it to get sh, and finally derives the one-time stealth address (Astealth) using Bob's public spending key.[13]
3. **Step 3: Asset Transfer.** The wallet constructs a standard transaction to send the desired asset (e.g., ETH, an ERC-20 token, or an NFT) directly to the newly generated Astealth. From the perspective of the EVM, this is an entirely normal transfer to a fresh EOA.[30]
4. **Step 4: On-Chain Announcement.** To ensure Bob can find the funds, Alice's wallet must publish the necessary recovery data. It does this by calling the announce function on the canonical ERC5564Announcer contract. This call emits the Announcement event containing the ephemeralPubKey (R), the stealthAddress (Astealth), and the metadata (including the view tag). This announcement can be bundled into the same transaction as the asset transfer or sent separately.[25]

5.3 The Recipient's Workflow: Scanning and Fund Recovery

This is the most computationally intensive and UX-challenging part of the process for the recipient, Bob.

1. **Step 1: Scanning for Announcements.** Bob's wallet must monitor the blockchain for every Announcement event emitted by the ERC5564Announcer contract. It typically

starts scanning from the block number of its last scan to avoid reprocessing old data.[19]

2. **Step 2: Filtering with the View Tag.** For each new Announcement, the wallet performs the initial, lightweight check. It takes the ephemeralPubKey from the event log, multiplies it by Bob's private viewing key (kview) to compute the shared secret point S, hashes it to get sh, and compares the first byte of sh with the viewTag from the event's metadata. If they do not match, the event is immediately discarded, and the wallet moves to the next one. This step filters out the vast majority of irrelevant transactions.[32]
3. **Step 3: Verifying Ownership.** In the rare case of a view tag match (a "false positive" or a genuine transaction), the wallet proceeds to the full verification. It uses the computed sh and Bob's public spending key (Pspend) to derive the full stealth public key (Pstealth) and its corresponding address. It then checks if this derived address matches the stealthAddress from the event log. If it matches, ownership is confirmed.[32]
4. **Step 4: Deriving the Private Key and Accessing Funds.** Once ownership is confirmed, the wallet performs the final step: it uses Bob's private *spending* key (kspend) and the shared secret scalar (sh) to compute the private key for the stealth address: kstealth=kspend+sh. This new private key is then imported into the wallet's key manager. Bob now has full control over the funds at the stealth address and can sign transactions to spend or transfer them as he would with any other account.[15]

Section 6: Practical Integration: Wallets, dApps, and Use Cases

The successful adoption of stealth addresses hinges on their seamless integration into the existing Web3 ecosystem. This requires thoughtful design from wallet developers and dApp builders to abstract the cryptographic complexity and provide a smooth, intuitive user experience. The standardization provided by ERC-5564 and ERC-6538 creates the foundation for this integration.

6.1 Wallet Integration Architecture

Integrating stealth address functionality into a crypto wallet involves several key architectural considerations that go beyond handling standard EOAs.

Secure Management of Spending and Viewing Keys

The dual-key model is central to the security of stealth addresses. Wallet developers must implement a robust key management system that reflects this separation of concerns.

- **Key Derivation:** The spending (kspend) and viewing (kview) keys should be deterministically derived from the user's master seed phrase, often by having the user sign a domain-specific message (e.g., "Sign this message to generate stealth keys for").[15] This ensures the keys are recoverable and tied to the user's single backup phrase.
- **Storage and Access Control:** The private spending key (kspend) must be treated with the highest level of security, equivalent to the master seed itself. It should ideally be stored in a secure enclave or hardware module and only accessed for the final step of deriving a stealth private key after a transaction has been confirmed. The private viewing key (kview), while still sensitive, can be stored in a more accessible "hot" state to facilitate frequent scanning operations.[41] The user interface should make it clear that exporting or exposing the viewing key allows others to see incoming payments but not spend them.

Designing Efficient Scanning Mechanisms

The most significant UX hurdle is the requirement to scan the blockchain for Announcement events. Wallets can adopt several strategies to manage this:

- **Client-Side Scanning:** The wallet itself can query an RPC node for all Announcement logs since the last scan and perform the cryptographic checks locally on the user's device. This offers maximum privacy as the user's keys never leave their device. However, it is resource-intensive, consuming significant bandwidth, CPU, and battery, making it challenging for mobile wallets or users with many incoming transactions.[19]
- **Delegated Scanning (Third-Party Service):** A more user-friendly approach involves a trusted third-party scanning service. The user provides their private viewing key (or a derived scanning key) to this service. The service scans the blockchain on the user's behalf and notifies the wallet when a potential transaction is found. The wallet can then perform the final verification and private key derivation locally. This vastly improves performance and reduces the burden on the client device but introduces a privacy trade-off: the scanning service learns of all the user's incoming stealth transactions.[15]
- **Hybrid Models:** A wallet could offer a hybrid approach, performing client-side scanning by default but allowing users to opt-in to a dedicated scanning provider for better performance. Future advancements in Private Information Retrieval (PIR) or Fully Homomorphic Encryption (FHE) may one day enable fully private, outsourced scanning, but these technologies are still in the research phase.[42]

UX/UI Considerations

A successful integration must hide the complexity from the end-user:

- **Sending:** The sending flow should be as simple as sending to an ENS name. The user enters the recipient's name, the wallet handles the registry lookup and cryptographic generation in the background, and the user simply signs a transaction.[15]
- **Receiving:** The wallet should present funds received at multiple stealth addresses in a unified, aggregated view. The initial scan upon opening the wallet may take time, and the UI should clearly communicate this process. Each stealth address with a balance should be manageable as a sub-account, with clear warnings about the privacy implications of withdrawing funds to a public, known address.[25]

6.2 Survey of Existing Implementations

Several projects have already built robust implementations of the ERC-5564 and ERC-6538 standards, serving as valuable case studies.

- **Umbra Cash:** Umbra is arguably the most well-known and widely used stealth address protocol on EVM chains. It provides a user-friendly dApp that demonstrates the full end-to-end flow. Users connect their wallet, sign a message to generate and register their spending and viewing keys on the ERC-6538 registry, and can then send and receive funds. Umbra's architecture relies on a combination of client-side scanning and a subgraph for fetching announcement data efficiently. It also integrates a relayer network to help solve the gas funding problem for ERC-20 token withdrawals.[18] It's worth noting that Umbra's specific
  secp256k1 implementation differs slightly from the one eventually included in the ERC-5564 specification, highlighting the importance of a finalized standard.[15]
- **Fluidkey:** Fluidkey is another implementation of the ERC-5564 standard that explores a different architectural path by integrating with smart contract wallets, specifically Gnosis Safe. This approach allows for the creation of stealth *Safes*, bringing the privacy benefits of stealth addresses to multi-signature accounts. This can be particularly useful for DAOs or teams that need to receive funds privately.[25]

6.3 Expanding the Design Space: dApp Use Cases

While private payments are the primary application, the ability to non-interactively generate a fresh, recipient-controlled address unlocks a wide range of innovative dApp use cases.

- **Confidential Donations and Public Goods Funding:** As previously discussed, non-profits, political campaigns, and open-source projects can use a single public stealth meta-address for fundraising. This protects the organization's financial privacy and prevents donor activity from being easily tracked, which could encourage larger or more frequent contributions.[10]
- **Private Payroll and Contractor Payments:** Businesses can leverage stealth addresses to pay employees and contractors without exposing their salary information on a public ledger. This protects employee privacy and prevents competitors from gaining intelligence on the company's operational costs and talent compensation.[10]
- **Anonymous NFT Airdrops and POAP Distribution:** Projects can airdrop NFTs or distribute Proof of Attendance Protocol (POAP) tokens to stealth addresses. This allows users to claim or receive these assets without creating a public on-chain link between their primary address and their participation in a specific event or community. This is particularly valuable for events where attendance might be sensitive information.[46]
- **Shielded DAO Voting:** In decentralized governance, voter anonymity can be crucial for preventing coercion or bribery. A DAO could grant voting rights by sending a governance token or a voting NFT to a member's stealth address. The member can then vote from this fresh, unlinked address, participating in governance without revealing their public identity's voting record.[25]

Section 7: A Comparative Analysis of EVM Privacy Solutions

Stealth addresses are a powerful tool, but they are just one of several approaches to on-chain privacy available in the EVM ecosystem. Understanding their specific trade-offs compared to alternatives like mixers and ZK-SNARK-based systems is essential for any developer choosing a privacy solution.

7.1 Stealth Addresses vs. Mixers (e.g., Tornado Cash)

- **Mechanism:** Mixers, such as the now-sanctioned Tornado Cash, operate by pooling assets from many different users into a large smart contract. A user deposits funds and receives a cryptographic note. At a later time, they can use this note to withdraw the same amount of funds to a completely new address. The protocol's goal is to break the on-chain link by ensuring that the withdrawn funds are indistinguishable from the funds of all other users in the pool.[10] In contrast, stealth addresses do not pool funds; they create a new, private destination for a peer-to-peer transfer.
- **Privacy Guarantees:** Stealth addresses provide strong *recipient anonymity*. The link between the transaction and the recipient's main identity is broken. However, the sender's address and the transaction amount remain public. Mixers provide *sender-recipient unlinkability*. They obscure the specific path of the funds, but the deposit and withdrawal actions themselves are public events. The privacy of a mixer depends heavily on the size of its anonymity set (the number of other users in the pool) and the user's operational security.
- **Trust Assumptions & Risks:** Stealth addresses are entirely non-custodial and cryptographically direct. The funds go from the sender to an EOA controlled by the recipient. The primary risk is user error (e.g., self-deanonymization). Mixers, while non-custodial in theory, have faced immense regulatory scrutiny. Using a mixer can lead to an address being flagged or blacklisted by centralized exchanges and other services, and the smart contracts themselves can be a point of failure or exploit.[25]

7.2 Stealth Addresses vs. ZK-SNARK-Based Shielded Pools (e.g., Railgun, Zcash)

- **Mechanism:** Systems like Railgun (on Ethereum) or the privacy coin Zcash utilize Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (ZK-SNARKs). These protocols maintain a separate, encrypted state within a smart contract (a "shielded pool"). When a user wants to transact privately, they generate a cryptographic proof that a valid state transition is occurring (e.g., "I own these notes and am creating new notes of equal value for another owner") without revealing any of the underlying details. The on-chain transaction only contains this proof, which the smart contract verifies.[25]
- **Privacy Guarantees:** ZK-SNARK-based systems offer the strongest form of privacy, achieving full confidentiality. They hide the sender, the receiver, and the transaction amount, making the transaction details opaque to all outside observers. This is a significant step up from stealth addresses, which only hide the recipient's link.
- **Complexity and Cost:** This superior privacy comes at a significant cost. Generating ZK-SNARK proofs is computationally intensive, requiring significant processing power on the user's device and leading to slower transaction creation times.[49] Furthermore, verifying these proofs on-chain consumes a substantial amount of gas, making ZK-based

transactions significantly more expensive than standard transfers. In comparison, stealth addresses are considered a "lightweight" approach. The cryptographic operations are relatively simple (a few scalar multiplications and hashes), and the on-chain footprint is minimal (a standard asset transfer plus an event emission), resulting in much lower gas costs and faster execution.[10]

Table: Comparison of EVM Privacy Technologies

The choice of a privacy technology depends on the specific requirements of the application, balancing the desired level of privacy with costs, complexity, and risks.

Technology	Privacy Guarantee	On-Chain Footprint	Gas Cost	Computational Cost	Key Risks
Stealth Addresses (ERC-5564)	Recipient Anonymity & Unlinkability	Standard transfer to a new EOA + one log event	Low to Medium	Low (Sender); High (Recipient Scanning)	Scanning overhead, gas funding for new addresses, user self-deanonymization.
Mixers (e.g., Tornado Cash)	Sender-Recipient Unlinkability	deposit and withdraw function calls and events	High	Low	Regulatory/blacklisting, small anonymity set risk, smart contract exploits.
ZK-SNARK Pools (e.g., Railgun)	Full Confidentiality (Sender, Receiver, Amount)	On-chain proof verification and state update	Very High	Very High (Proof Generation); High (Verification)	Cryptographic complexity, potential for bugs in complex

					circuits, usability challenges.

Section 8: Challenges, Limitations, and the Frontier of Stealth Address Research

While stealth addresses offer a compelling and relatively lightweight solution for on-chain privacy, their practical implementation is not without significant challenges. These hurdles span computational efficiency, user experience, and operational security, and are the focus of ongoing research and development in the field.

8.1 The Scanning Problem: Computational Overhead and Scalability

The most formidable obstacle to the widespread adoption of stealth addresses is the scanning problem.[53] To discover incoming funds, a recipient's wallet must process every single

Announcement event emitted by the ERC5564Announcer contract since its last check.[17] This constitutes a linear scan of on-chain data. As the number of stealth address users and transactions grows, the number of events to process will increase proportionally.

For a client-side wallet, particularly on a mobile device, this presents a severe scalability challenge. Each check involves fetching event logs and performing cryptographic computations, consuming significant bandwidth, CPU cycles, and battery life.[54] A user who receives frequent payments or has been offline for an extended period might face a prohibitively long and resource-intensive sync time upon opening their wallet. This computational burden makes the user experience cumbersome and pushes the architecture towards reliance on centralized or trusted third-party scanning services, which reintroduces a degree of privacy erosion.[20]

8.2 Optimization Techniques: View Tags and Beyond

The architects of ERC-5564 were acutely aware of the scanning problem, and the primary optimization built into the standard is the **view tag**. As detailed previously, this one-byte filter allows a wallet to discard approximately 99.6% (255/256) of irrelevant announcements using only a single scalar multiplication and a hash, avoiding the more expensive operations.[32] While a crucial improvement, this is still a mitigation, not a solution; the wallet must still process every event to check the tag.

The research frontier is actively exploring more advanced solutions to overcome the linear scan requirement:

- **Elliptic Curve Pairings:** Certain stealth address protocols, like ECPDKSAP (Elliptic Curve Pairing Dual-Key Stealth Address Protocol), leverage the properties of bilinear pairings on elliptic curves. These can be constructed to allow for more efficient batch verification or other cryptographic tricks that can speed up the scanning process significantly compared to the standard Diffie-Hellman-style approach (DKSAP) used in the ERC-5564 reference implementation.[22]
- **Advanced Cryptographic Primitives:** Researchers are investigating novel schemes that shift the computational burden. **Fuzzy Message Detection** allows a user to provide a "fuzzy" key to a server, which can filter messages with a defined false-positive rate.[58] More advanced concepts like
  Oblivious Message Retrieval (OMR) use techniques like Fully Homomorphic Encryption (FHE) to allow a user to privately query a database of announcements without revealing to the server which announcement they are interested in. While promising, these methods currently have significant trade-offs, such as large key sizes or heavy server-side computation.[20]

8.3 The Gas Funding Dilemma

A stealth address generated via ERC-5564 is a fresh Externally Owned Account (EOA). When it receives an asset, such as an ERC-20 token or an NFT, its native ETH balance is zero.[17] To perform any action with that asset—like transferring it, selling it, or interacting with a dApp—the user must pay transaction fees (gas) in ETH.

This creates a critical privacy dilemma. If the user funds the new stealth address with ETH from one of their known, public wallets, they create a direct, permanent on-chain link between their public identity and the stealth address, completely nullifying the privacy benefits.[10] Several solutions and workarounds exist to address this:

- **Sender Sponsorship:** The simplest solution is for the sender to include a small amount

of ETH along with the primary asset transfer. This "gas stipend" provides the recipient with the necessary funds to make their first transaction from the stealth address without compromising their privacy.[17]

- **Relayers and Meta-Transactions:** The recipient can sign an off-chain message that authorizes a specific action (e.g., "transfer my 100 DAI to address X"). This signed message can be passed to a third-party relayer. The relayer submits the transaction to the blockchain and pays the gas fee in ETH. To be compensated, the relayer can be paid via a smart contract that allows them to take a small portion of the ERC-20 tokens being transferred. Implementations like Umbra have integrated this functionality.[19]
- **Account Abstraction (ERC-4337):** The rise of Account Abstraction offers a more native and elegant solution. If stealth addresses were implemented as smart accounts instead of EOAs, they could leverage the ERC-4337 ecosystem. This would allow a "Paymaster" contract to sponsor the gas fee for the stealth account's transaction, with the Paymaster being reimbursed in the ERC-20 token being transferred, all within a single, atomic transaction.

8.4 Privacy Hygiene: Preventing Self-Deanonymization

Technology alone cannot guarantee privacy; user behavior is a critical component. Even with a perfectly implemented stealth address system, users can inadvertently deanonymize themselves through poor privacy hygiene.

- **Consolidating Funds:** The most common mistake is transferring funds from multiple, separate stealth addresses into a single, new address. This action creates a public on-chain record that links all those stealth addresses together via the common-input-ownership heuristic, suggesting they are controlled by the same entity.[25]
- **Withdrawing to Known Addresses:** Similarly, withdrawing funds from a stealth address directly to a KYC'd centralized exchange account or a public ENS address immediately and irrevocably links that stealth address to the user's real-world identity.[10]
- **Best Practices:** To maintain privacy, funds in a stealth address should ideally be spent directly from that address or moved through a privacy-preserving protocol (like a mixer or ZK-pool) before being sent to a non-private destination.

8.5 Future Outlook: Post-Quantum Stealth Addresses

The security of the current stealth address implementation relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem. A sufficiently powerful quantum computer running

Shor's algorithm could theoretically break this cryptography, compromising the entire system. Recognizing this long-term threat, researchers are already exploring the design of post-quantum stealth address protocols. These schemes would replace ECC with quantum-resistant cryptographic primitives, such as those based on the Learning With Errors (LWE) problem from lattice-based cryptography, ensuring that the privacy guarantees remain secure in a post-quantum world.[55]

The challenges inherent in stealth address technology reveal a deeper truth: its success is not contingent on the perfection of its core cryptography alone. Rather, its practical viability is deeply interconnected with the evolution of the broader Ethereum infrastructure. The scanning problem highlights the need for dedicated infrastructure layers or advanced cryptographic solutions for private data retrieval. The gas funding problem points directly to the necessity of robust relayer networks and, more fundamentally, the native gas abstraction capabilities of ERC-4337. Even the cost of making announcements on-chain is a limiting factor, a problem that could be mitigated by cheaper data availability layers like EIP-4844's blobspace. Therefore, stealth addresses will not mature in a vacuum; they will co-evolve with and be enabled by the surrounding ecosystem of scaling, abstraction, and data solutions.

Section 9: Strategic Recommendations for Developers and Protocols

The successful integration and adoption of stealth addresses require a concerted effort from developers across the ecosystem, from wallet providers to dApp builders. By following best practices and adopting standardized architectural patterns, developers can abstract the system's complexity and deliver its privacy benefits to end-users effectively.

9.1 Best Practices for Integrating ERC-5564 into Wallets

For wallet developers, the primary goal is to provide a user experience that is both secure and intuitive, hiding the intricate mechanics of stealth addresses from the user.

- **Architectural Separation of Keys:** Implement a clear and secure separation between the spending and viewing keys derived from the user's master seed. The spending key must be afforded the highest level of protection, while the viewing key should be accessible for scanning operations. The wallet's architecture should enforce this separation to prevent accidental exposure of the spending key.[15]

- **Hybrid Scanning Model:** Offer users a flexible scanning solution. A default client-side scanning mode should be available for maximum privacy. Additionally, provide an opt-in feature to connect to a trusted (or user-specified) scanning provider for improved performance, especially on mobile devices. The UI must clearly communicate the privacy trade-offs of using a third-party service.[15]
- **Intuitive User Interface:** The UI should abstract away the concept of multiple addresses. It should present a unified balance for all stealth funds and provide a clear transaction history. Crucially, the wallet must include prominent warnings when a user attempts an action that could compromise their privacy, such as sending funds from a stealth address to their public ENS address or consolidating funds from multiple stealth addresses.[41]

9.2 Architectural Patterns for Building Stealth-Enabled dApps

For dApp developers, integrating stealth addresses can unlock new privacy-preserving features and use cases.

- **Leverage the Standards:** Build directly on the ERC-5564 and ERC-6538 standards. Use libraries and SDKs that implement the canonical contracts for the Announcer and Registry. This ensures interoperability with the entire ecosystem of stealth-address-enabled wallets.[15]
- **Registry for Discovery:** For dApps that need to send assets to users (e.g., airdrops, rewards, POAPs), the workflow should involve looking up the user's stealth meta-address from the ERC6538Registry. This provides a seamless experience where the user only needs to provide their ENS name or EOA.[30]
- **Integrate Gas Solutions:** To address the gas funding dilemma for users receiving non-native tokens, dApps should consider integrating with relayer services or building support for ERC-4337 paymasters. This allows users to pay for withdrawal transactions using the tokens they've just received, removing a major friction point and privacy risk.[63]

9.3 Considerations for Protocol-Level Privacy Implementations

While application-layer privacy solutions like the current ERC-5564 standard are a significant step forward, the most robust privacy is achieved when it is a native, default feature of the protocol itself. The long-term vision for Ethereum privacy should involve exploring ways to integrate stealth address-like functionality at a lower level. This could mean making stealth addresses a native account type or building privacy-preserving features directly into the

transaction layer. By making privacy the default, rather than an opt-in feature, the anonymity set includes all users of the network, providing much stronger guarantees and fulfilling the original vision of a truly private, peer-to-peer electronic cash system on a global, decentralized scale.[10] The ongoing work on account abstraction and protocol-level improvements provides a pathway toward this more private future.

Sources des citations

1. www.investopedia.com, consulté le septembre 18, 2025, https://www.investopedia.com/terms/s/stealth-address-cryptocurrency.asp#:~:text=Stealth%20addresses%20are%20a%20technique,be%20traced%20on%20the%20blockchain.
2. Blockchain Data Privacy Concerns - Identity Management Institute®, consulté le septembre 18, 2025, https://identitymanagementinstitute.org/blockchain-data-privacy-concerns/
3. Stealth Addresses and Web3 transaction privacy | Chainstack Blog, consulté le septembre 18, 2025, https://chainstack.com/stealth-addresses-blockchain-transaction-privacy/
4. Privacy and blockchain - Wikipedia, consulté le septembre 18, 2025, https://en.wikipedia.org/wiki/Privacy_and_blockchain
5. Stealth Address (Cryptocurrency): Meaning and Concerns - Investopedia, consulté le septembre 18, 2025, https://www.investopedia.com/terms/s/stealth-address-cryptocurrency.asp
6. Blockchain for marketing? Maybe, but privacy issues abound - MIT Sloan, consulté le septembre 18, 2025, https://mitsloan.mit.edu/ideas-made-to-matter/blockchain-marketing-maybe-privacy-issues-abound
7. Stealth Address – Blockchain Patterns - CSIRO Research, consulté le septembre 18, 2025, https://research.csiro.au/blockchainpatterns/general-patterns/stealth-address/
8. Blockchain Security: Common Issues & Vulnerabilities | NordLayer, consulté le septembre 18, 2025, https://nordlayer.com/blog/blockchain-security-issues/
9. How Blockchain and AI Enable Personal Data Privacy and Support Cybersecurity - PDXScholar, consulté le septembre 18, 2025, https://pdxscholar.library.pdx.edu/cgi/viewcontent.cgi?article=1236&context=busadmin_fac
10. Stealth Addresses Explained (And How To Use Them) - Nekodex by Perpetual Protocol, consulté le septembre 18, 2025, https://perpprotocol.mirror.xyz/DieQewuZGzAhFjBOkr3AbhZsqwVbZf6WIxh9axxL9gA
11. Blockchain Security: Preventing Threats Before They Strike - Chainalysis, consulté le septembre 18, 2025, https://www.chainalysis.com/blog/blockchain-security/
12. Stealth Address | Moneropedia | Monero - secure, private, untraceable, consulté le septembre 18, 2025, https://www.getmonero.org/resources/moneropedia/stealthaddress.html

13. What are stealth addresses, and how do they work? - Cointelegraph, consulté le septembre 18, 2025, https://cointelegraph.com/explained/what-are-stealth-addresses-and-how-do-they-work
14. SoFi Becomes First US Bank to Adopt Bitcoin Lightning - OWNR Wallet, consulté le septembre 18, 2025, https://ownrwallet.com/blog/what-are-stealth-addresses-and-how-do-they-work-ownr-wallet/
15. Stealth Addresses Introduction - GitHub Pages, consulté le septembre 18, 2025, https://nerolation.github.io/stealth-utils/
16. What is Stealth Address technology and Why Does Monero Use It? - SerHack, consulté le septembre 18, 2025, https://serhack.me/articles/what-is-stealth-address-technology-monero/
17. An incomplete guide to stealth addresses, consulté le septembre 18, 2025, https://vitalik.eth.limo/general/2023/01/20/stealth.html
18. Private Transactions on Ethereum using Stealth Addresses (ERC-5564) | QuickNode Guides, consulté le septembre 18, 2025, https://www.quicknode.com/guides/ethereum-development/wallets/how-to-use-stealth-addresses-on-ethereum-eip-5564
19. Frequently Asked Questions - Umbra Cash, consulté le septembre 18, 2025, https://app.umbra.cash/faq
20. Anonymity Analysis of the Umbra Stealth Address Scheme on Ethereum - arXiv, consulté le septembre 18, 2025, https://arxiv.org/pdf/2308.01703
21. Stealth addresses 101 – PraneshASP | Blog, consulté le septembre 18, 2025, https://flawsomedev.com/blog/stealth-addresses-101
22. Elliptic Curve Pairing Stealth Address Protocols - arXiv, consulté le septembre 18, 2025, https://arxiv.org/html/2312.12131v2
23. ECDH address - Bitcoin Wiki, consulté le septembre 18, 2025, https://en.bitcoin.it/wiki/ECDH_address
24. How do stealth addresses work? - Bitcoin - Reddit, consulté le septembre 18, 2025, https://www.reddit.com/r/Bitcoin/comments/2p3qno/how_do_stealth_addresses_work/
25. Privacy in Ethereum — Stealth Addresses | by Simon Brown - Medium, consulté le septembre 18, 2025, https://simbro.medium.com/privacy-in-ethereum-stealth-addresses-f05016109010
26. Key derivation function - Wikipedia, consulté le septembre 18, 2025, https://en.wikipedia.org/wiki/Key_derivation_function
27. Stealth Address and Key Management Techniques in Blockchain Systems - SciTePress, consulté le septembre 18, 2025, https://www.scitepress.org/papers/2017/62700/62700.pdf
28. Stealth Address - Cyphertalk, consulté le septembre 18, 2025, https://muens.io/stealth-address/
29. ScopeLift/stealth-address-erc-contracts: Contracts for ERC-5564 Stealth

Addresses and ERC-6538 Stealth Meta-Address Registry - GitHub, consulté le septembre 18, 2025, https://github.com/ScopeLift/stealth-address-erc-contracts
30. kassandraoftroy/erc5564-contracts: ERC5564 smart contracts for stealth addresses protocols on evm chains - GitHub, consulté le septembre 18, 2025, https://github.com/kassandraoftroy/erc5564-contracts
31. BaseSAP: Modular Stealth Address Protocol for Programmable Blockchains - arXiv, consulté le septembre 18, 2025, https://arxiv.org/pdf/2306.14272
32. ERC-5564 Stealth Addresses - Ethereum Magicians, consulté le septembre 18, 2025, https://ethereum-magicians.org/t/erc-5564-stealth-addresses/10614
33. ERC-5564 Stealth Addresses - #10 by iAmMichaelConnor - Ethereum Magicians, consulté le septembre 18, 2025, https://ethereum-magicians.org/t/erc-5564-stealth-addresses/10614/10
34. [ERC5564] 送受信者のみがアクセスできるステルスアドレスの仕組みを理解しよう！ - Qiita, consulté le septembre 18, 2025, https://qiita.com/cardene/items/115df838afdb3f582731
35. ERC- 6538 - EIPs Insights, consulté le septembre 18, 2025, https://eipsinsight.com/ercs/erc-6538
36. Stealth Meta-Address Registry - EIPs - Fellowship of Ethereum Magicians, consulté le septembre 18, 2025, https://ethereum-magicians.org/t/stealth-meta-address-registry/12888
37. ERC6538Registry | Address 0x6538E6bf4B0eBd30A8Ea093027Ac2422ce5d6538 - OP Sepolia Scan, consulté le septembre 18, 2025, https://sepolia-optimism.etherscan.io/address/0x6538E6bf4B0eBd30A8Ea093027Ac2422ce5d6538
38. Private Payments and Stealth Addresses | by Specter Protocol - Medium, consulté le septembre 18, 2025, https://medium.com/@specter-eth/private-payments-and-stealth-addresses-0ea28f6b34ce
39. Stealth Address Donations | ETHGlobal, consulté le septembre 18, 2025, https://ethglobal.com/showcase/stealth-address-donations-5hk7v
40. ScopeLift/umbra-protocol: Privacy Preserving Shielded Payments On The Ethereum Blockchain - GitHub, consulté le septembre 18, 2025, https://github.com/ScopeLift/umbra-protocol
41. zkFi: Privacy-Preserving and Regulation Compliant Transactions using Zero Knowledge Proofs June 2023 - arXiv, consulté le septembre 18, 2025, https://arxiv.org/html/2307.00521v4
42. Anonymity Analysis of the Umbra Stealth Address Scheme on Ethereum - ResearchGate, consulté le septembre 18, 2025, https://www.researchgate.net/publication/372832303_Anonymity_Analysis_of_the_Umbra_Stealth_Address_Scheme_on_Ethereum
43. zkStealth - Submissions, consulté le septembre 18, 2025, https://projects.ethberlin.org/teams/948
44. Umbra v2: Flexible, Interoperable, and More than Just Payments - ScopeLift, consulté le septembre 18, 2025, https://scopelift.co/blog/introducing-umbra-v2-architecture

45. Stealth Addresses - ErgoDocs, consulté le septembre 18, 2025, https://docs.ergoplatform.com/uses/stealth-address/
46. POAPrivacy | ETHGlobal, consulté le septembre 18, 2025, https://ethglobal.com/showcase/poaprivacy-ykk5a
47. Enabling Privacy Transactions Through Stealth Addresses | by REI Network - Medium, consulté le septembre 18, 2025, https://medium.com/gxchain-project/enabling-privacy-transactions-through-stealth-addresses-68fdb4951f85
48. Vitalik Buterin proposes stealth addresses for anonymous NFT ownership - Cointelegraph, consulté le septembre 18, 2025, https://cointelegraph.com/news/vitalik-buterin-proposes-stealth-addresses-for-anonymous-nft-ownership
49. What Is A Privacy Coin | Examples Of Privacy Coins - Skrill, consulté le septembre 18, 2025, https://www.skrill.com/en/crypto/the-skrill-crypto-academy/advanced/what-is-a-privacy-coin/
50. Stealth Addresses & Shielded Pools | Bitget News, consulté le septembre 18, 2025, https://www.bitget.com/news/detail/12560603813738
51. What Are Privacy Coins? Guide to Private Crypto in 2025 - Crypto News, consulté le septembre 18, 2025, https://cryptonews.com/academy/what-are-privacy-coins/
52. Privacy Coins Will Make You Untraceable! 6 Best Privacy Coins For 2025 - Transak, consulté le septembre 18, 2025, https://transak.com/blog/privacy-coins
53. Privacy Protection Method for Blockchain Transactions Based on the Stealth Address and the Note Mechanism - MDPI, consulté le septembre 18, 2025, https://www.mdpi.com/2076-3417/14/4/1642
54. Open problem: improving stealth addresses - Cryptography - Ethereum Research, consulté le septembre 18, 2025, https://ethresear.ch/t/open-problem-improving-stealth-addresses/7438
55. More Efficient Stealth Address Protocol - arXiv, consulté le septembre 18, 2025, https://arxiv.org/html/2504.06744v1
56. (PDF) More Efficient Stealth Address Protocol - ResearchGate, consulté le septembre 18, 2025, https://www.researchgate.net/publication/390638945_More_Efficient_Stealth_Address_Protocol
57. Elliptic Curve Pairing Stealth Address Protocols - arXiv, consulté le septembre 18, 2025, https://arxiv.org/pdf/2312.12131
58. Is there a way to optimize a linear scan while preserving anonymity?, consulté le septembre 18, 2025, https://crypto.stackexchange.com/questions/30148/is-there-a-way-to-optimize-a-linear-scan-while-preserving-anonymity
59. Introducing Umbra – Privacy Preserving Stealth Payments On The Ethereum Blockchain, consulté le septembre 18, 2025, https://scopelift.co/blog/introducing-umbra
60. (PDF) Post-Quantum Stealth Address Protocols - ResearchGate, consulté le septembre 18, 2025,

https://www.researchgate.net/publication/388354033_Post-Quantum_Stealth_Address_Protocols
61. More Efficient Stealth Address Protocol - arXiv, consulté le septembre 18, 2025, https://arxiv.org/pdf/2504.06744
62. Post-Quantum Stealth Address Protocols - arXiv, consulté le septembre 18, 2025, https://arxiv.org/pdf/2501.13733
63. Umbra v2 Prototypes And Designs - ScopeLift, consulté le septembre 18, 2025, https://scopelift.co/blog/umbra-v2-prototypes-and-designs