Stealth and Beyond: Attribute-Driven Accountability in Bitcoin Transactions

Alberto Maria Mongardini[1⋆], Daniele Friolo[1⋆], and Giuseppe Ateniese[2]

[1] Department of Computer Science, Sapienza University of Rome, Rome, Italy
{mongardini, friolo}@di.uniroma1.it
[2] Department of Computer Science, George Mason University, Fairfax, VA, USA
ateniese@gmu.edu

Abstract. Bitcoin enables decentralized, pseudonymous transactions, but balancing privacy with accountability remains a challenge. This paper introduces a novel dual accountability mechanism that enforces both sender and recipient compliance in Bitcoin transactions. Senders are restricted to spending Unspent Transaction Outputs (UTXOs) that meet specific criteria, while recipients must satisfy legal and ethical requirements before receiving funds. We enhance stealth addresses by integrating compliance attributes, preserving privacy while ensuring policy adherence. Our solution introduces a new cryptographic primitive, Identity-Based Matchmaking Signatures (IB-MSS), which supports streamlined auditing. Our approach is fully compatible with existing Bitcoin infrastructure and does not require changes to the core protocol, preserving both privacy and decentralization while enabling transaction auditing and compliance.

1 Introduction

Bitcoin has transformed financial transactions by enabling pseudonymous, decentralized exchanges without intermediaries. This model empowers users to manage assets autonomously and conduct global transactions freely. However, as cryptocurrency adoption grows, so too does the tension between *privacy* and *regulatory compliance*—particularly with Anti-Money Laundering (AML) and Know Your Customer (KYC) requirements. While Bitcoin offers pseudonymity, its transparent public ledger exposes transaction histories to public scrutiny, making it increasingly difficult to preserve user privacy while adhering to legal standards.

To address privacy concerns, *stealth addresses* [14,25,30] were introduced. These allow recipients to generate one-time-use addresses that are not directly linked to their public Bitcoin addresses. This obfuscation ensures that third parties cannot easily trace the recipient of a transaction, providing a layer of

⋆ A. M. Mongardini, D. Friolo —The work was carried out whilst the authors were visiting George Mason University, Fairfax, VA, USA. The two authors contributed equally.

privacy by breaking the linkability of transactions. Stealth addresses thus help preserve recipient anonymity within Bitcoin's inherently transparent system.

However, while stealth addresses enhance privacy, they fall short in addressing the equally critical need for *accountability*. They do not provide mechanisms to ensure that transactions comply with legal or ethical standards, nor can they enforce policy-based criteria in complex regulatory environments. For example:

– **Donations to organizations in restricted regions**: NGOs operating in politically sensitive regions may need to ensure that donations come from legitimate, legally compliant sources to avoid funding from illicit entities. Traditional stealth addresses ensure donor privacy but lack the ability to verify that the funds originate from vetted and legally approved sources. This leaves the NGO vulnerable to unknowingly accepting illegal donations, exposing it to serious legal risks.
– **Compliance in cross-border remittances**: Individuals sending remittances across borders often face regulatory scrutiny, especially in cases involving countries under sanctions or financial restrictions. For example, sending funds to a sanctioned country could violate international regulations. While stealth addresses can maintain privacy by hiding recipient details, they provide no way to enforce compliance with such regulations, potentially leading to unlawful transactions.
– **Ensuring AML compliance for cryptocurrency exchanges**: Cryptocurrency exchanges must comply with AML/KYC regulations, which require them to validate the identities of both senders and recipients. Traditional stealth addresses obscure participants for privacy but do not allow for built-in verification of AML/KYC compliance, leaving exchanges struggling to balance regulatory adherence with user privacy.
– **Private investments in regulated markets**: Investors participating in private transactions, such as venture capital investments or equity crowdfunding, often require anonymity for various reasons (e.g., protecting strategic financial interests). However, these transactions must comply with securities laws, which require verification of accredited investor status or legal financial limits. Traditional stealth addresses provide anonymity but lack mechanisms to ensure that only accredited or verified investors participate in these transactions.

In response to these challenges, we propose an enhanced form of stealth addresses that goes beyond privacy protection to incorporate accountability. Our solution integrates a novel cryptographic primitive (*Identity-Based Matchmaking Signatures*) into stealth addresses, embedding compliance attributes directly within the transaction. This ensures that both *privacy* and *accountability* are preserved.

Specifically, our approach introduces a *dual accountability* mechanism:

1. **Sender Accountability**: The recipient can only spend funds if the sender has met specific criteria, such as originating from a permitted country or complying with AML regulations.

2. **Recipient Accountability**: The sender can ensure that the recipient satisfies legal or ethical requirements, preventing the transfer of funds to sanctioned or restricted entities.

Our solution integrates seamlessly with the existing Bitcoin infrastructure, requiring no changes to the Bitcoin Core protocol. The only addition is a *Certification Authority (CA)*, which issues compliance certificates to users, verifying that both senders and recipients meet the necessary standards. The CA can be implemented in centralized or decentralized formats, depending on the regulatory context or user preference.

Our contributions. This paper introduces a comprehensive framework for enhancing accountability in Bitcoin transactions through the use of *stealth addresses with attributes*, underpinned by a novel cryptographic primitive, *Identity-Based Matchmaking Signatures* (IB-MSS). Our key contributions are as follows:

– We introduce the *IB-MSS* primitive, which integrates dual accountability attributes into stealth addresses for both the sender and the recipient. Inspired by Matchmaking Encryption (ME) [6], where decryption requires both parties' attributes to satisfy each other's policies, our IB-MSS adapts this concept to signatures. This enables signature verification only when both the sender's and recipient's attributes fulfill each other's specified policies, ensuring dual accountability.
  Building upon non-interactive key distribution [20] and digital signature schemes, we present the detailed construction of IB-MSS in Section 4.
– We explore the integration of a *Certification Authority (CA)* to validate sender and recipient attributes. We discuss centralized and decentralized implementations of this authority in Section 4.3.
– We provide, in Section 5, an implementation of stealth addresses with attributes using IB-MSS and benchmark its performance. Our implementation is inspired by Cerulli *et al.*'s work on verifiably encrypted threshold key derivation [10].

Moreover, in Section 2.1, we also introduce potential enhancements, such as time-locked stealth addresses and stealth transactions with embedded messages.

1.1 Notation

Let $\mathbb{N}$ be the set of all natural numbers; for $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We use calligraphic letters to denote sets (such as $\mathcal{X}$). Throughout the paper, we use the abbreviation PPT to refer to probabilistic polynomial time. Given a PPT algorithm A, we denote by $\mathsf{A}(x)$ the probability distribution of the output of A when run on input x. If algorithm A has oracle access to a set of oracles $\mathcal{O}$, we denote by $\mathcal{Q}_O$ the set of queries made by A to an oracle $O \in \mathcal{O}$.

We use the symbol $:=$ to assign the output value of the algorithm on the right-hand side to the variable on the left-hand side. When x is chosen uniformly from a set $\mathcal{X}$, we write $x \leftarrow\!\!\$\, \mathcal{X}$. If A is an algorithm, we write $y \leftarrow\!\!\$\, \mathsf{A}(x)$ to denote a run of A on input x with output y and random tape $r \leftarrow\!\!\$\, \{0,1\}^\lambda$.

We denote the security parameter by $\lambda \in \mathbb{N}$, and an arbitrary positive polynomial by $\mathsf{poly}(\cdot)$. Every algorithm takes the security parameter λ as input (in unary, i.e., 1^λ). When an algorithm has multiple inputs, 1^λ is typically omitted. A function $\nu : \mathbb{N} \to \mathbb{R}$ is called negligible if, for every positive polynomial $\mathsf{poly}(\cdot)$ and for all sufficiently large λ, it holds that $\nu(\lambda) \leq \frac{1}{\mathsf{poly}(\lambda)}$. We use $\mathsf{negl}(\lambda)$ to denote an unspecified negligible function. In the remainder of the paper, we consider the function $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ as a type-1 pairing.

2 Stealth addresses

Bitcoin's public ledger makes all transactions visible, exposing user activity to analysis. This compromises the pseudonymity of traditional addresses, as they can be traced back to individuals. Stealth addresses counter this by breaking the visible link between sender and recipient.

Basic Stealth Address Protocol (BSAP) Introduced in 2011 [14], stealth addresses create temporary, one-time-use public addresses to enhance transaction privacy. For simplicity, we use multiplicative notation. Let $\mathbb{G}$ be a group of prime order p with generator g. Alice (the sender) and Bob (the receiver) have public/secret key pairs $(\mathsf{pk_A}, \mathsf{sk_A})$ and $(\mathsf{pk_B}, \mathsf{sk_B})$, where $\mathsf{pk_A} = g^a$, $\mathsf{pk_B} = g^b$, $\mathsf{sk_A} = a$, and $\mathsf{sk_B} = b$ for $a, b \leftarrow\!\!\$\, \mathbb{Z}_p$. Using the Diffie-Hellman protocol, they establish a shared secret k:

- $k = H(g^{\mathsf{sk_A}\mathsf{sk_B}}) = H(g^{ab}) = H(\mathsf{pk_B^a}) = H(\mathsf{pk_A^b})$,
- Alice uses g^k as the recipient's address,
- Bob spends funds sent to g^k using k as his private key.

While this protocol provides privacy by generating one-time-use addresses, the static nature of the address reduces privacy over multiple transactions. Both parties also share the ability to compute the private key, which limits security.

Improved Stealth Address Protocol (ISAP) Proposed in 2013 [25] and adopted in Bitcoin in 2014 [30], ISAP improves upon BSAP by introducing more dynamic addresses:

- Alice generates a fresh key pair (r, R), where $R = g^r$ and $r \leftarrow\!\!\$\, \mathbb{Z}_p$. She computes $k = H(\mathsf{pk_B^r}) = H(g^{rb}) = H(\mathsf{R}^b)$,
- She computes $\mathsf{Q} = g^k \cdot \mathsf{pk_B}$ and creates a transaction with Q and embeds R (either in `OP_RETURN` or as part of the transaction signature's randomness [34]) to ensure stealthiness,
- Bob monitors the blockchain for transactions containing R', computes $k' = H(\mathsf{R}'^b)$, and reconstructs the recipient address $\mathsf{Q}' = g^{k'} \cdot \mathsf{pk_B}$. If $\mathsf{Q}' = \mathsf{Q}$, he can spend the transaction using $k' + b$.

While ISAP improves privacy by creating fresh addresses, Bob must use his private spending key to monitor the blockchain, which introduces a security risk. Although Bob does not directly share his private spending key with any third party, actively using it for blockchain scanning increases the risk of exposure. Indeed, repeated use of his sensitive key makes it vulnerable to potential threats such as software bugs, memory leaks, or malware.

Dual Key Stealth Address Protocol (DKSAP) It mitigates ISAP's security concerns by introducing two separate key pairs for tracking and spending:

- Bob holds two key pairs: a tracking key pair $(\mathsf{ptk_B}, \mathsf{stk_B})$ with $\mathsf{stk_B} = s$ and $\mathsf{ptk_B} = g^s$, and a spending key pair $(\mathsf{pk_B}, \mathsf{sk_B})$,
- Alice generates (r, R) and computes $k = H(\mathsf{ptk_B}^r)$,
- She calculates $\mathsf{Q} = g^k \cdot \mathsf{pk_B}$ and creates a transaction with Q and R,
- Bob monitors the blockchain using the tracking key $\mathsf{stk_B}$ to find R', computes $k' = H(\mathsf{R}'^s)$, and reconstructs $\mathsf{Q}' = g^{k'} \cdot \mathsf{pk_B}$. If $\mathsf{Q}' = \mathsf{Q}$, Bob can spend the transaction using his spending key.

By separating the tracking and spending keys, Bob can outsource transaction monitoring without compromising his private spending key, enhancing both security and privacy.

2.1 Enhancements of Stealth Addresses

While stealth addresses enhance privacy in Bitcoin, our work extends their functionality with additional cryptographic features. In this subsection, we introduce our enhancements to stealth addresses, which serve as an intermediate step toward presenting our main contribution. We integrate concepts such as Identity-Based Encryption (IBE) and time-lock mechanisms, enabling the inclusion of hidden information within stealth addresses to support more advanced use cases.

A practical application of this is the creation of time-locked stealth transactions, where cryptographic puzzles are embedded in the shared key k. These puzzles ensure that the recipient can only spend the transaction after a certain time has passed. For example:

$$k = H(\mathsf{pk_B}^r, \text{“101101”}) = H(g^{br}, \text{“101101”}).$$

where $\mathsf{pk_B}$ is the recipient's public key, r is a random value chosen by the sender, and g^{br} is the Diffie-Hellman shared key. The value “101101” serves as a time-lock condition.

Unlike traditional time-lock methods, where the lock time is visible, this approach embeds the time constraint within the stealth address. Only the recipient, who computes k, can identify the time-lock, preserving privacy while preventing censorship due to visible lock times.

To further refine time control, we can use a puzzle based on repeated squaring [24]:

$$H(\mathsf{pk}_{\mathsf{B}}^{r}, 2^{e}) = H(g^{br}, 2^{e}),$$

where $e = 2^t \mod \phi(n)$, with t representing the desired lock time, and $n = pq$ being the product of two large primes p and q chosen by Alice. This ensures that only after time t can the recipient unlock and spend the funds.

The same cryptographic puzzle string used in time-locked addresses can double as a covert message, ensuring confidentiality. This dual-use of k for time-locking and discreet communication allows only the intended recipient to decode and access the message, such as "42", for example:

$$k = H(\mathsf{pk}_{\mathsf{B}}^{r}, \text{“42”}) = H(g^{br}, \text{“42”}).$$

Incorporating IBE into k adds further flexibility by allowing identity-specific transactions. In IBE, a sender encrypts data so that only a recipient matching a specific identity (e.g., an email address) can decrypt it. The receiver needs a decryption key tied to their identity, issued by a trusted authority.

In the Boneh-Franklin IBE scheme [8], the trusted authority generates a decryption key for the identity id as $H'(\mathsf{id})^s$, where s is the authority's secret key. The sender, knowing the recipient's identity id and the authority's master public key $\mathsf{mpk} = g^s$, encrypts the message as follows:

$$c = (g^t, m \oplus e(H'(\mathsf{id}), g^s)^t),$$

where t is a random value. The recipient, with the decryption key $H'(\mathsf{id})^s$, can retrieve the message m by using the pairing properties:

$$e(H'(\mathsf{id})^s, g^t) = e(H'(\mathsf{id}), g^s)^t.$$

By adapting this IBE structure, the shared key k can be modified as follows:

$$k = H(\mathsf{pk}_{\mathsf{B}}^{r}, e(H'(\mathsf{id}), g^s)^r) = H(g^{br}, e(H'(\mathsf{id})^s, g^r)).$$

This adaptation allows the sender to restrict the transaction to a specific recipient identity, providing additional security and control. In the context of IBE, the identity id can represent not just simple identifiers like email addresses, but also complex attributes or conditions required for the recipient to access the funds. For example, id could be defined as id = "manager $\wedge$ after Jan 2028", meaning the recipient must be a manager and can only access the funds after January 2028. In this case, the trusted authority issues the decryption key only when both conditions are met. This attribute-based approach extends the flexibility of stealth addresses, enabling conditional access based on roles, time constraints, or other attributes, making them suitable for sophisticated use cases.

This solution requires no changes to Bitcoin's Core or transaction structure. The recipient runs a program, similar to current stealth address handling, to identify transactions they can decrypt. The only addition is an external authority for issuing decryption keys when the conditions are satisfied. This ensures enhanced privacy and accountability without disrupting standard transaction processes.

3 Related Works

Considerable progress has been made in formalizing stealth addresses and implementing efficient solutions for Bitcoin and Ethereum.

Formalization. Stealth addresses have seen significant formalization efforts, beginning with the work of Fleischhacker *et al.* [13], who introduced Digital Signatures with Re-Randomizable Keys. This model established a strong unforgeability notion, allowing adversaries to query a key re-randomization oracle while maintaining security.

Meiklejohn and Mercer [19] proposed stealth keys, where a shared secret and nonce allow one-time public and private key derivation, with indistinguishability guarantees between derived and freshly generated keys. Fauzi *et al.* [11] extended this model to support Updatable Public Keys, ensuring that public keys could be refreshed without revealing whether they were newly generated or updated, further strengthening the privacy of stealth addresses.

Backes *et al.* [7] introduced Signatures with Flexible Keys, allowing users to transform public keys into equivalent forms, while maintaining indistinguishability, even when an adversary has access to the randomness used in key generation.

Liu *et al.* [16] identified a vulnerability in deterministic wallets that could expose the master secret key if a single one-time key were compromised. They proposed Key-Insulated and Privacy-Preserving Signatures (PDPKS) to address key exposure risks. This work was extended by Pu *et al.* [23], who introduced Stealth Signatures, providing stronger protection against key exposure and offering a post-quantum secure construction. Our IB-MSS design builds on these advancements, particularly the formalization of stealth addresses.

A recent approach to enhance recipient privacy is the Silent Payments protocol [15]. It operates similarly to stealth addresses but differs in how the one-time-use address is constructed. Instead of using her private key sk_A, Alice uses the private key corresponding to one of the UTXOs used in the payment. This eliminates the need for the nonce R required in ISAP, ensuring that the stealth address remains dynamic. While Silent Payments remove the need for R, the recipient still needs to monitor the blockchain to identify transactions sent to them. Like the original stealth address protocols, Silent Payments use the Diffie-Hellman protocol. Thus, the enhancements we introduce in Sec. 2.1, as well as the Accountable Stealth Address with Attributes protocol (Sec. 4), can also be applied to Silent Payments.

Implementation. The development of stealth addresses for blockchains has also progressed. Wahrstätter *et al.* [33] introduced the BaseSap protocol, designed for programmable blockchains, using the Secp256k1 elliptic curve. Their implementation leverages view tags to optimize transaction parsing, enabling more efficient handling of stealth addresses in decentralized applications.

Recent privacy-focused efforts, such as Privacy Pools by Buterin *et al.* [9], proposed in response to sanctions on Tornado Cash [31], further highlight the need for privacy-preserving systems that allow users to prove compliance with legal frameworks. Privacy Pools aim to enable users to demonstrate that their funds are not connected to illicit activities while maintaining privacy. However, challenges in preventing illegal transactions through privacy tools remain [27].

4 Accountable Stealth Addresses with Attributes

Stealth addresses protect identities but cannot enforce *who* may pay *whom*. We close that gap by folding a short *attribute certificate*—for instance "citizen of X"—directly into the Diffie–Hellman-style secret that a stealth address already uses. No extra bytes appear on–chain; the policy is evaluated inside the shared secret k.

Running example. Bob accepts coins only from citizens of X, whereas Alice will pay only citizens of Y. A certification authority (CA) issues long-lived credentials $C_X = H_1(X)^s$ and $C_Y = H_1(Y)^s$, where s is the CA's master secret and H_1 hashes its input into the pairing group.

Creating the payment. Before funding the output, Alice computes

$$k = H_2\big(e(C_X,\, H_1(Y))\big)$$

and derives a one-time public key from k; the resulting script is an ordinary SegWit `OP_0 <20-byte hash>` (P2WPKH).

Spending it. When Bob scans the chain, he evaluates

$$k = H_2\big(e(H_1(X),\, C_Y)\big).$$

The two values coincide only if *both* policies hold; otherwise, the output is either invisible (to Bob) or unspendable (to Alice).

Discouraging credential resale. A raw certificate such as $H_1(\text{citizen\_of } X)^s$ is portable. To curb resale, the CA can apply one of two *off-chain* restrictions that leave the on-chain script unchanged:

- **Hardware anchoring.** The certificate is sealed in a "no-export" secure element (*e.g.*, FIDO2 token [18], Trusted Platform Module (TPM) [21], or phone enclave [5,28]). The chip may also store a private NotAfter date and/or a small usage counter; once either limit is reached, it refuses further calls. Reselling, therefore, requires handing over the physical token, which stops working after its internal quota or deadline.
- **Optional time-window rotation.** If hardware anchoring is unavailable, the CA can simply re-publish the common "citizen-of-X" certificate on a fixed schedule (*e.g.*, once per month). A leaked key then dies automatically when the window closes. This measure *only shortens the resale lifetime*; the original owner could still leak the next month's key, so it should be viewed as a stop-gap rather than a complete anti-resale solution.

Formalization. The scheme instantiates an *Identity-Based Matchmaking Stealth Signature* (IB-MSS), adapted from the stealth-signature framework of Pu *et al.* [23]. IB-MSS provides correctness, unlinkability and unforgeability, while the on-chain transaction remains byte-for-byte identical to a standard SegWit P2WPKH output.

4.1 Identity-Based Matchmaking Stealth Signatures

We define a new primitive called Identity-Based Matchmaking Stealth Signatures (IB-MSS). In IB-MSS, a sender generates a one-time public key using a retrieval key related to their identity, obtained from a Public Key Generator (PKG), and coupled with the receiver's identity. Symmetrically, the receiver can recover the one-time secret key related to the same one-time public key by using their retrieval key, also obtained from the PKG, and coupled with the sender's identity. The formalization of this concept is inspired by the former work on Matchmaking Encryption by Ateniese *et al.* [6]. We follow a similar algorithmic structure.

We formalize two main properties: unforgeability and unlinkability. Unforgeability ensures that no external adversary can forge a signature verifiable from such a one-time public key. Unlinkability ensures that no external adversary can distinguish between a public key generated from a standard digital signature scheme's key generation algorithm and a one-time public key generated from the IB-MSS, even if the respective one-time secret key is leaked.

The description of IB-MSS follows. **We implicitly assume that all algorithms after KGen and before Sign take mpk as input.**

An IB-MSS scheme Π with message space $\mathcal{M}$ is composed of a tuple of algorithms $(\mathsf{MKGen}, \mathsf{KGen}, \mathsf{SKGen}, \mathsf{RKGen}, \mathsf{OPKGen}, \mathsf{OSKGen}, \mathsf{Sign}, \mathsf{Vrfy})$, described as follows:

$\mathsf{MKGen}(1^\lambda)$**:** On input the security parameter 1^λ, the randomized master key generator outputs a master public key mpk, and a master secret key msk.

$\mathsf{KGen}(1^\lambda)$**:** On input the security parameter 1^λ, the randomized key generation algorithm outputs a public/private key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{SKGen}(\mathsf{msk}, \sigma)$**:** On input the master secret key msk and a sender identity $\sigma \in \{0,1\}^*$, the sender key generator outputs a sender retrieval key srk.

$\mathsf{RKGen}(\mathsf{msk}, \rho)$**:** On input the master secret key msk and a receiver identity $\rho \in \{0,1\}^*$, the receiver key generator outputs a receiver retrieval key rrk.

$\mathsf{OPKGen}(\mathsf{srk}, \mathsf{pk}, \rho)$**:** On input a sender retrieval key srk, the receiver public key pk and a receiver identity $\rho \in \{0,1\}^*$, the one-time public key generator outputs a one-time public key opk.

$\mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma)$**:** On input a one-time public key opk, a receiver retrieval key rrk, a receiver secret key sk and a sender identity $\sigma \in \{0,1\}^*$, the one-time secret key generator outputs a one-time secret key osk, or $\perp$ indicating failure.

$\mathsf{Sign}(\mathsf{osk}, m)$**:** On input a one-time secret key osk and a message $m \in \mathcal{M}$, the randomized signing algorithm outputs a signature τ.

$\mathsf{Vrfy}(\mathsf{opk}, m, \tau)$**:** On input a one-time public key opk, a message $m \in \mathcal{M}$, and a signature τ, the verification algorithm outputs 1 (valid) or 0 (invalid).

Definition 1 (Correctness). *An Identity-Based Matchmaking Stealth Signature scheme is* correct *if, for every security parameter λ, every message $m \in \mathcal{M}$, and every identities $\sigma, \rho \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{opk}, m, \tau) = 1 \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{MKGen}(1^\lambda); \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda); \\ \mathsf{srk} := \mathsf{SKGen}(\mathsf{msk}, \sigma); \\ \mathsf{rrk} := \mathsf{RKGen}(\mathsf{msk}, \rho); \\ \mathsf{opk} := \mathsf{OPKGen}(\mathsf{srk}, \mathsf{pk}, \rho); \\ \mathsf{osk} := \mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma); \\ \tau \leftarrow\$ \mathsf{Sign}(\mathsf{osk}, m) \end{array}\right] = 1.$$

Definition 2 (Unforgeability). *An Identity-Based Matchmaking Stealth Signature scheme Π satisfies existential unforgeability w.r.t. chosen message attacks if, for all valid PPT adversaries* A, $\Pr\left[\mathsf{EUFCMA}^{\Pi}_{\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Let $\mathcal{O} = \{\mathsf{SKGen}(\mathsf{msk}, \cdot), \mathsf{RKGen}(\mathsf{msk}, \cdot), \mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}\}$*. The experiment proceeds as follows:*

```
$\mathsf{EUFCMA}^{\Pi}_{\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{MKGen}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$\mathcal{Q}_{\mathsf{Sign}\mathcal{O}} := \emptyset$
$\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \emptyset$
$(i^*, m^*, \tau^*) \leftarrow\$ \mathsf{A}^{\mathcal{O}}(\mathsf{mpk}, \mathsf{pk})$
$(\mathsf{opk}^*, \mathsf{osk}^*, \cdot, \sigma^*, \rho^*) := \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}[i^*]$
$b_0 := \mathsf{Vrfy}(\mathsf{opk}^*, m^*, \tau^*) = 1$
$b_1 := (i^*, m^*) \notin \mathcal{Q}_{\mathsf{Sign}\mathcal{O}}$
$b_2 := (\mathsf{opk}^*, \mathsf{osk}^*, \mathsf{true}, \sigma^*, \rho^*) \notin \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$
return $b_0 \wedge b_1 \wedge b_2$
```

```
$\mathsf{OSKGen}\mathcal{O}(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)$
$\mathsf{rrk} := \mathsf{RKGen}(\mathsf{msk}, \rho)$
$\mathsf{osk} := \mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma)$
if $\mathsf{opk} = \mathsf{pk}$ then $\mathsf{osk} := \mathsf{sk}$
$\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} \cup \{(\mathsf{opk}, \mathsf{osk}, \mathsf{flag}, \sigma, \rho)\}$
if $\mathsf{flag} = \mathsf{true}$ then return $\mathsf{osk}$
else return 1
```

```
$\mathsf{Sign}\mathcal{O}(i, m)$
$(\mathsf{opk}, \mathsf{osk}) := \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}[i]$
$\tau \leftarrow\$ \mathsf{Sign}(\mathsf{osk}, m)$
$\mathcal{Q}_{\mathsf{Sign}\mathcal{O}} := \mathcal{Q}_{\mathsf{Sign}\mathcal{O}} \cup \{(i, m)\}$
return $\tau$
```

The adversary A *is valid if one of the following two conditions is satisfied:*

- *(Unforgeability w.r.t the attributes) for all $\sigma \in \mathcal{Q}_{\mathsf{SKGen}(\mathsf{msk},\cdot)}$, $\rho \in \mathcal{Q}_{\mathsf{RKGen}(\mathsf{msk},\cdot)}$, and all $(i, m) \in \mathcal{Q}_{\mathsf{Sign}\mathcal{O}}$ we have that, given $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}[i] = (\cdot, \cdot, \cdot, \sigma', \rho')$, it holds that $\sigma' \neq \sigma$, $\rho' \neq \rho$. Moreover, whenever $(\cdot, \cdot, \cdot, \sigma', \rho') \in \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$ we must have $(\cdot, \cdot, \cdot, \rho', \sigma') \notin \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.*
- *(Unforgeability w.r.t the public key)* $(\cdot, \cdot, \mathsf{true}, \cdot, \cdot) \notin \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.

Definition 3 (Unlinkability). *An Identity-Based Matchmaking Stealth Signature scheme Π satisfies* unlinkability *if, for all valid PPT adversaries* $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$, $\Pr\left[\mathsf{UNLINK}^{\Pi}_{\mathsf{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Let $\mathcal{O} = \{\mathsf{SKGen}(\mathsf{msk}, \cdot), \mathsf{RKGen}(\mathsf{msk}, \cdot), \mathsf{OSKGen}\mathcal{O}, \mathsf{Sign}\mathcal{O}\}$. $\mathsf{UNLINK}^{\Pi}_{\mathsf{A}}(\lambda)$ *follows:*

```
$\mathsf{UNLINK}^{\Pi}_{\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{MKGen}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$
$b \leftarrow_\$ \{0, 1\}$
$\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \emptyset$
$(\sigma^*, \rho^*, \mathsf{st}) \leftarrow_\$ \mathsf{A}_0^{\mathcal{O}}(\mathsf{mpk}, \mathsf{pk})$
$(\mathsf{opk}_0, \mathsf{osk}_0) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$
$\mathsf{srk} := \mathsf{SKGen}(\mathsf{msk}, \sigma^*)$
$\mathsf{rrk} := \mathsf{RKGen}(\mathsf{msk}, \rho^*)$
$\mathsf{opk}_1 := \mathsf{OPKGen}(\mathsf{srk}, \mathsf{pk}, \rho^*)$
$\mathsf{osk}_1 := \mathsf{OSKGen}(\mathsf{opk}_1, \mathsf{rrk}, \mathsf{sk}, \sigma^*)$
$b' \leftarrow_\$ \mathsf{A}_1^{\mathcal{O}}(\mathsf{opk}_b, \mathsf{st})$
return $b \overset{?}{=} b'$
```

```
$\mathsf{OSKGen}\mathcal{O}(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)$
$\mathsf{rrk} := \mathsf{RKGen}(\mathsf{msk}, \rho)$
$\mathsf{osk} := \mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma)$
if $\mathsf{opk} = \mathsf{pk}$ then $\mathsf{osk} := \mathsf{sk}$
if $\mathsf{opk} = \mathsf{opk}_b$ then $\mathsf{osk} := \mathsf{osk}_b$
$\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} \cup \{(\mathsf{opk}, \mathsf{osk}, \mathsf{flag}, \sigma, \rho)\}$
if $\mathsf{flag} = \mathsf{true}$ then return $\mathsf{osk}$
else return $1$
```

```
$\mathsf{Sign}\mathcal{O}(b^*, i, m)$
if $i = -1$ then
  $\tau \leftarrow_\$ \mathsf{Sign}(\mathsf{osk}_b, m)$
else
  $(\mathsf{opk}, \mathsf{osk}, \cdot, \cdot, \cdot) = \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}[i]$
  $\tau \leftarrow_\$ \mathsf{Sign}(\mathsf{osk}, m)$
return $\tau$
```

The adversary A *is valid if one of the following two conditions is satisfied:*

– *(Unlinkability w.r.t. attributes) for all* $\sigma' \in Q_{\mathsf{SKGen}}$ *and* $\rho' \in Q_{\mathsf{RKGen}}$ *we have* $\sigma' \neq \sigma^* \neq \rho'$ *and* $\sigma' \neq \rho^* \neq \rho'$*; additionally* $(\cdot, \cdot, \cdot, \sigma^*, \rho^*) \notin Q_{\mathsf{OSKGen}\mathcal{O}}$ *and* $(\cdot, \cdot, \cdot, \rho^*, \sigma^*) \notin Q_{\mathsf{OSKGen}\mathcal{O}}$.
– *(Unlinkability w.r.t. the public key)* $(\cdot, \cdot, \mathsf{true}, \cdot, \cdot) \notin \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.

4.2 IB-MSS from IB-NIKD

We now show how to instantiate IB-MSS from any Identity-Based Non-Interactive Key Distribution (IB-NIKD) scheme as defined by Paterson and Srinivasan [20], in combination with an EUF-CMA signature scheme where $\mathsf{pk} = g^{\mathsf{sk}}$, g is a generator of a group $\mathbb{G}$ of order q, and $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_q$.

IB-NIKD Definition [20]. Let $\mathcal{SHK}$ be a shared key space. An IB-NIKD scheme Π is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{SharedKey})$ described as follows:

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter, this randomized algorithm outputs a master public key mpk and a master secret key msk.

$\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$ **:** On input the master public key mpk, the master secret key msk, and an identifier $\mathsf{id} \in \{0, 1\}^*$, outputs a private key $\mathsf{rk}_{\mathsf{id}}$.

$\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_{\mathsf{id}_A}, \mathsf{id}_B)$**:** On input the master public key mpk, a private key $\mathsf{rk}_{\mathsf{id}_A}$, and an identifier $\mathsf{id}_B \in \{0,1\}^*$, outputs a shared key $k \in \mathcal{SHK}$.

Definition 4 (Correctness). *An IB-NIKD is correct if,* $\forall \mathsf{id}_A, \mathsf{id}_B \in \{0,1\}^*$,

$$\Pr[\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_{\mathsf{id}_A}, \mathsf{id}_B) = \mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_{\mathsf{id}_B}, \mathsf{id}_A)] = 1,$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{Setup}(1^\lambda)$, $\mathsf{rk}_{\mathsf{id}_A} \leftarrow_\$ \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_A)$, *and* $\mathsf{rk}_{\mathsf{id}_B} \leftarrow_\$ \mathsf{Extract}$ $(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_B)$.

Definition 5 (IND-Security). *An IB-NIKD scheme Π is IND-secure if, for all valid PPT adversaries* A, $\Pr\left[\mathsf{IND\text{-}SK}^{\mathsf{A}}_{\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Let $\mathcal{O} = \{\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \cdot), \mathsf{Reveal}\mathcal{O}\}$. $\mathsf{IND\text{-}SK}^{\mathsf{A}}_{\Pi}(\lambda)$ *follows:*

```
$\mathsf{IND\text{-}SK}^{\Pi}_{\mathsf{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{Setup}(1^\lambda)$
$(\mathsf{id}_A, \mathsf{id}_B, \mathsf{st}) \leftarrow_\$ \mathsf{A}^{\mathcal{O}}(\mathsf{mpk})$
$b \leftarrow_\$ \{0,1\}$
$\mathsf{rk}_{\mathsf{id}_A} \leftarrow_\$ \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_A)$
if $b = 0$
    $k = \mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_{\mathsf{id}_A}, \mathsf{id}_B)$
else
    $k \leftarrow_\$ \mathcal{SHK}$
$b' \leftarrow_\$ \mathsf{A}^{\mathcal{O}}(\mathsf{mpk}, \mathsf{id}_A, \mathsf{id}_B, k, \mathsf{st})$
return $b \overset{?}{=} b'$
```

```
$\mathsf{Reveal}\mathcal{O}(\mathsf{id}_{A^*}, \mathsf{id}_{B^*})$
$\mathsf{rk}_{\mathsf{id}_{A^*}} \leftarrow_\$ \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_{A^*})$
return $\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_{\mathsf{id}_{A^*}}, \mathsf{id}_{B^*})$
```

The adversary A *is valid if, for all* $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{Extract}(\mathsf{mpk},\mathsf{msk},\cdot)}$, $\mathsf{id}^* \neq \mathsf{id}_A \wedge \mathsf{id}^* \neq \mathsf{id}_B$, *and* $(\mathsf{id}_A, \mathsf{id}_B) \notin \mathcal{Q}_{\mathsf{Reveal}\mathcal{O}}$ *and* $(\mathsf{id}_B, \mathsf{id}_A) \notin \mathcal{Q}_{\mathsf{Reveal}\mathcal{O}}$.

EUF-CMA Signature Scheme. A signature scheme is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ with message space $\mathcal{M}$, described as follows:

$\mathsf{KGen}(1^\lambda)$**:** On input the security parameter, outputs a verification key/signing key pair $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$**:** On input a signing key sk and a message $m \in \mathcal{M}$, outputs a signature τ.

$\mathsf{Vrfy}(\mathsf{vk}, m, \tau)$**:** On input a verification key vk and a message $m \in \mathcal{M}$, outputs 1 if the signature verifies, and 0 otherwise.

Definition 6 (Correctness). *A signature scheme Π is correct if, for all* $m \in \mathcal{M}$, $\Pr[\mathsf{Vrfy}(\mathsf{vk}, m, \tau) = 1]$, *where* $(\mathsf{vk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$ *and* $\tau \leftarrow_\$ \mathsf{Sign}(\mathsf{sk}, m)$.

Definition 7 (EUF-CMA). *A signature scheme Π is existentially unforgeable under chosen message attacks if, for all valid PPT adversaries* A,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \tau) = 1 \ \wedge m \notin \mathcal{Q}_{\mathsf{Sign}}(\mathsf{sk}, \cdot) \,\middle|\, \begin{array}{c}(\mathsf{vk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(1^\lambda); \\ (m, \tau) \leftarrow_\$ \mathsf{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Let Π_{SIG} be an EUF-CMA signature scheme with message space $\mathcal{M}$, and assume $\mathsf{KGen}(1^\lambda)$ outputs $(\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk} \leftarrow_\$ \mathbb{Z}_q)$, where $\mathbb{G}$ is a group of order q. Let Π_{NIKD} be an Identity-Based Non-Interactive Key Distribution scheme with key space $\mathcal{SHK} = \mathbb{Z}_q$. We construct our IB-MSS scheme Π_{IBMSS} as follows:

$\mathsf{MKGen}(1^\lambda)$**:** Output $(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \Pi_{\mathsf{NIKD}}.\mathsf{Setup}(1^\lambda)$.
$\mathsf{KGen}(1^\lambda)$**:** Output $(\mathsf{pk}(= g^{\mathsf{sk}}), \mathsf{sk}) \leftarrow_\$ \Pi_{\mathsf{SIG}}.\mathsf{KGen}(1^\lambda)$.
$\mathsf{SKGen}(\mathsf{msk}, \mathsf{id})$ **and** $\mathsf{RKGen}(\mathsf{msk}, \mathsf{id})$**:** Output $\Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$.
$\mathsf{OPKGen}(\mathsf{srk}, \mathsf{pk}, \rho)$**:** Set $k = \Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{srk}, \rho)$ and output $g^k \cdot \mathsf{pk}$.
$\mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma)$**:** Set $k = \Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rrk}, \sigma)$. If $g^{k+\mathsf{sk}} = \mathsf{opk}$, output $k + \mathsf{sk}$, else output $\perp$.
$\mathsf{Sign}(\mathsf{osk}, m)$**:** Output $\Pi_{\mathsf{SIG}}.\mathsf{Sign}(\mathsf{osk}, m)$.
$\mathsf{Vrfy}(\mathsf{opk}, m, \tau)$**:** Output $\Pi_{\mathsf{SIG}}.\mathsf{Vrfy}(\mathsf{opk}, m, \tau)$.

In the following theorem, whose detailed proof appears in Appendix A, we state that Π_{IBMSS} satisfies the security properties we require.

Theorem 1. *Assuming that Π_{NIKD} is IND-secure and Π_{SIG} is EUF-CMA secure, the IB-MSS scheme Π_{IBMSS} described above satisfies correctness, unforgeability, and unlinkability.*

Proof (Sketch).
Unforgeability. The proof is divided into the two adversarial-validity subcases. For the *attribute* subcase we introduce two hybrids Hyb_0 and Hyb_1. In Hyb_1, for a uniformly random query index i^*, the key pair $(\mathsf{opk}_{i^*}, \mathsf{osk}_{i^*})$ is generated with $\Pi_{\mathsf{SIG}}.\mathsf{KGen}(1^\lambda)$ instead of the $\mathsf{SharedKey}$ procedure. Hybrids Hyb_0 and Hyb_1 are indistinguishable by the IND security of Π_{NIKD} and the EUF-CMA security of Π_{SIG}. Any successful forgery in Hyb_1 therefore transfers to a forgery against Π_{SIG}.

For the *public-key* subcase we give a direct reduction to the EUF-CMA security of Π_{SIG} (instantiated as Schnorr).

Unlinkability. For attributes, distinguishing the challenge bit would violate the IND security of Π_{NIKD}. For the public-key branch we observe that $(\mathsf{opk}_1, \mathsf{osk}_1)$ can be generated by KGen; a hybrid argument shows the resulting public key distribution is identical in both worlds.

Correctness follows immediately from the correctness of Π_{NIKD} (matching shared keys) and Π_{SIG} (valid signatures).

4.3 Certification Authority

In our protocol, a Certification Authority (CA) ensures that participants comply with specific regulations, such as AML/KYC requirements, while preserving privacy. The CA facilitates secure generation of shared secrets between parties based on certified attributes, enabling both privacy and regulatory compliance. We examine two CA models: centralized and decentralized, both constructed using our IB-MSS scheme instantiated from any IB-NIKD Π_{NIKD} (Section 4.2).

Centralized CA A centralized CA, such as a major exchange (e.g., Coinbase or Kraken), performs Know Your Customer (KYC) procedures to verify user identities for compliance. The CA issues certificates that confirm specific attributes, such as user identity or location, after users complete KYC.

Consider the case where Alice and Bob, both users of a centralized exchange C, have completed KYC and now wish to transact. Alice must satisfy attribute σ, and Bob must satisfy attribute ρ. The CA C operates as follows:

- C holds a master secret key msk and publishes a master public key mpk.
- Alice requests a private key (certificate) for her attribute σ, and C computes $D_A = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \sigma)$, providing it to Alice.
- Bob similarly requests his certificate $D_B = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \rho)$ from C.

Decentralized CA In a decentralized setup, protocols like VetKeys [10], developed by DFINITY, enable secure key derivation and policy compliance without relying on centralized authorities. VetKeys supports on-chain key derivation and verification, allowing users to comply with policies through decentralized applications (DApps). The process, depicted in Fig. 1, proceeds as follows:

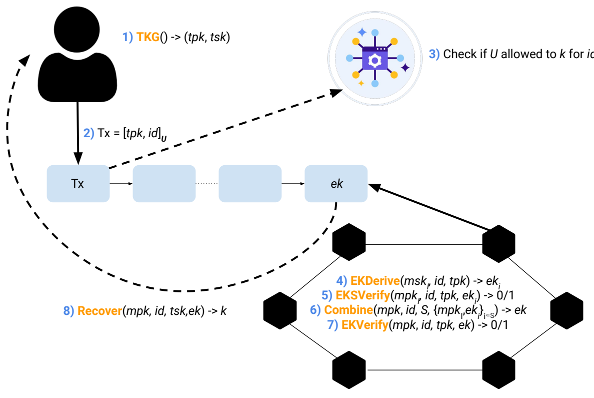
1) TKG() -> (tpk, tsk)
3) Check if U allowed to k for id
2) Tx = [tpk, id]_U
Tx
ek
4) EKDerive(msk_i, id, tpk) -> ek_i
5) EKSVerify(mpk_i, id, tpk, ek_i) -> 0/1
6) Combine(mpk, id, S, {mpk_i,ek_i}_{i∈S}) -> ek
7) EKVerify(mpk, id, tpk, ek) -> 0/1
8) Recover(mpk, id, tsk,ek) -> k

Fig. 1. VetKeys scheme in the blockchain context.

- Alice generates a transport key pair $(\mathsf{tpk}_A, \mathsf{tsk}_A)$ and submits her transport public key tpk_A along with her policy attribute σ to a DApp D.
- The DApp verifies Alice's compliance with the policy σ (e.g., using zero-knowledge proofs or on-chain credentials).
- If compliant, the DApp nodes execute the VetKeys protocol to generate an encrypted derived key ek_A, which is stored on the blockchain.
- Alice retrieves ek_A and, using her transport secret key tsk_A, recovers her derived key $D_A = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \sigma)$.
- Bob similarly obtains his derived key $D_B = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \rho)$.

Shared Secret Computation In both centralized and decentralized CA models, Alice and Bob use their derived keys to compute a shared secret. They compute the same one-time stealth address secret key osk using $\Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, D_A, \rho)$ and $\Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, D_B, \sigma)$ respectively.

This ensures compliance and privacy in both models.

5 Implementation and Experiments

We aim to illustrate the practicality of our proposed approach within the Bitcoin ecosystem, highlighting its capability to achieve transaction accountability while preserving user privacy. To this end, we present a comprehensive performance evaluation of our implementation, which is based on the instantiation of our IB-MSS (Section 4.2) from the Sakai-Ohgishi-Kasahara [26] IB-NIKD and Schnorr signatures [29]. We describe our instantiantion together with a detailed analysis of parsing time costs in Bitcoin in Appendix B.

The proof of concept is implemented using Python 3.10.10, leveraging Charm 0.50 [3]. Our pairing-based scheme is instantiated using the SS512 curve.The experiments were conducted on a machine equipped with an Intel Core i7-12700 CPU (2.10GHz) and 32GB of RAM, running ManjaroLinux 22.1.0. The source code for our implementation is available on Anonymous GitHub [4].

Operation	**Minimum** (ms)	**Average** (ms)	**Maximum** (ms)
Setup	1.228	1.250	1.298
SKGen	1.942	1.996	2.139
RKGen	1.951	1.995	2.155
Enc	1.741	1.823	1.916
RetrieveKey	1.738	1.762	1.911

Table 1. Time costs in milliseconds for computing cryptographic functions.

Table 1 shows the time costs (in milliseconds) of the primary cryptographic operations. We conducted 100 independent runs for each operation, repeating each experiment ten times to extract minimum, average, and maximum execution times. For the main cryptographic operations, such as Enc and RetrieveKey, the average execution times were 1.823 ms and 1.762 ms, respectively.

To further demonstrate the feasibility of our protocol, we executed two accountable stealth transactions on the Bitcoin Testnet. These transactions were performed in both centralized [1] and decentralized [2] settings, using a modified version of the VetKeys protocol for the decentralized scenario.

Our protocol ensures that no additional information is embedded in transactions, making them indistinguishable from standard Bitcoin transactions. As a result, the transaction fees remain equivalent to those of standard Pay-to-Witness-Public-Key-Hash (P2WPKH) transactions. P2WPKH, introduced with the Segregated Witness (SegWit) update through BIP141 [17], was chosen over Pay-to-Public-Key-Hash (P2PKH) due to its lower transaction fees [12].

6 Conclusions and Open Problems

This paper introduced "stealth addresses with attributes," an innovative extension of stealth address protocols that enhances accountability while preserving privacy in Bitcoin transactions. By embedding compliance attributes into the stealth address framework, we achieve both sender and recipient accountability without compromising anonymity. The integration of Identity-Based Matchmaking Signatures (IB-MSS) further ensures verifiable compliance within Bitcoin's existing infrastructure.

However, two main challenges remain. The first is the computational overhead of parsing and verifying transactions with embedded attributes, which impacts scalability as transaction volumes grow. Future work should focus on optimizing parsing and verification processes to handle larger loads efficiently. The second challenge involves implementing practical and scalable zero-knowledge proofs for recipient accountability within the IB-MSS framework. Addressing this is essential to ensuring the protocol remains efficient and secure in real-world applications.

References

1. Blockstream explorer (2023), https://blockstream.info/testnet/tx/e65a99f0a260300c010270c35636d0497c2a89eb07bc04aa148fb71b076f1408
2. Blockstream explorer (2023), https://blockstream.info/testnet/tx/b5e74da90c37fdfbaafc9afa6a845147cbff53ae9818d1426ffb27b16ab936f4
3. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering **3**, 111–128 (2013), https://api.semanticscholar.org/CorpusID:2876079
4. anonStealth: Accountable stealth transactions with attributes. https://anonymous.4open.science/r/Attribute-Driven-Accountability-in-Bitcoin-Transactions-AD70/README.md (2023)
5. Apple: Secure Enclave (2025), https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web
6. Ateniese, G., Francati, D., Nuñez, D., Venturi, D.: Match me if you can: Matchmaking encryption and its applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 701–731. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_24
7. Backes, M., Hanzlik, L., Kluczniak, K., Schneider, J.: Signatures with flexible public key: Introducing equivalence classes for public keys. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 405–434. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_14
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13

9. Buterin, V., Illum, J., Nadler, M., Schär, F., Soleimani, A.: Blockchain privacy and regulatory compliance: Towards a practical equilibrium. Available at SSRN (2023)
10. Cerulli, A., Connolly, A., Neven, G., Preiss, F.S., Shoup, V.: vetkeys: How a blockchain can keep many secrets. Cryptology ePrint Archive (2023)
11. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 649–678. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_23
12. FixedFloat: Bitcoin address formats and performance comparison (2022), https://fixedfloat.com/en/blog/guides/bitcoin-address-formats
13. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 301–330. Springer, Berlin, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49384-7_12
14. Forum, B.: Bytecoin (2011), https://bitcointalk.org/index.php?topic=199.msg1670#msg1670
15. josibake: Silent payments (2023), https://github.com/bitcoin/bips/blob/master/bip-0352.mediawiki
16. Liu, Z., Yang, G., Wong, D.S., Nguyen, K., Wang, H.: Key-insulated and privacy-preserving signature scheme with publicly derived public key. In: IEEE EuroS&P. pp. 215–230. IEEE (2019)
17. Lombrozo, E.: Segregated witness (consensus layer) (2015), https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki#p2wpkh
18. Lyastani, S.G., Schilling, M., Neumayr, M., Backes, M., Bugiel, S.: Is fido2 the kingslayer of user authentication? a comparative usability study of fido2 passwordless authentication. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 268–285. IEEE (2020)
19. Meiklejohn, S., Mercer, R.: Möbius: Trustless tumbling for transaction privacy. PoPETs **2018**(2), 105–121 (Apr 2018). https://doi.org/10.1515/popets-2018-0015
20. Paterson, K.G., Srinivasan, S.: On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. Des. Codes Cryptogr. **52**(2), 219–241 (2009). https://doi.org/10.1007/S10623-009-9278-Y, https://doi.org/10.1007/s10623-009-9278-y
21. Perez, R., Sailer, R., van Doorn, L., et al.: vtpm: virtualizing the trusted platform module. In: Proc. 15th Conf. on USENIX Security Symposium. pp. 305–320 (2006)
22. Privacy, S.F.: View tags: How one byte will reduce monero wallet sync times by 40https://localmonero.co/knowledge/view-tags-reduce-monero-sync-time
23. Pu, S., Thyagarajan, S.A.K., Döttling, N., Hanzlik, L.: Post quantum fuzzy stealth signatures and applications. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 371–385. ACM (2023). https://doi.org/10.1145/3576915.3623148, https://doi.org/10.1145/3576915.3623148
24. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto (1996), https://people.csail.mit.edu/rivest/pubs/RSW96.pdf
25. van Saberhagen, N.: Cryptonote v 2.0 (2013), https://cryptonote.org/whitepaper.pdf
26. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. The 2000 Symposium on Cryptography and Information Security pp. 26–28 (2000)

27. Salvo, M.D.: Tornado cash user 'dusts' hundreds of public wallets—including celebs jimmy fallon, steve aoki and logan paul (2022), https://decrypt.co/107090/tornado-cash-dusts-public-wallets-jimmy-fallon-brian-armstrong-steve-aoki-logan-paul?amp=1
28. Samsung: Knox Vault (2025), https://docs.samsungknox.com/admin/fundamentals/whitepaper/samsung-knox-mobile-security/system-security/knox-vault/
29. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989). https://doi.org/10.1007/0-387-34805-0_22, https://doi.org/10.1007/0-387-34805-0_22
30. Todd, P.R.: Stealth addresses in bitcoin., https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
31. TREASURY, U.D.O.T.: U.s. treasury sanctions notorious virtual currency mixer tornado cash (2022), https://home.treasury.gov/news/press-releases/jy0916
32. UkoeHB: Reduce scan times with 1-byte-per-output 'view tag' (2020), https://github.com/monero-project/research-lab/issues/73
33. Wahrstätter, A., Solomon, M., DiFrancesco, B., Buterin, V., Svetinovic, D.: Basesap: Modular stealth address protocol for programmable blockchains. arXiv preprint arXiv:2306.14272 (2023)
34. Wiki, B.: ECDH Addresses, https://en.bitcoin.it/wiki/ECDH_address, [Online; accessed 13-Jun-2023]

A Proof of Theorem 1

Proof. We will prove that the IB-MSS scheme Π_{IBMSS} satisfies correctness, unlinkability, and unforgeability under the assumptions that Π_{NIKD} is IND-secure and Π_{SIG} is EUF-CMA secure.

Correctness. By the correctness of Π_{NIKD}, for any sender identity σ and receiver identity ρ, the shared keys computed by both parties are equal:

$$k = \Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{srk}, \rho) + \mathsf{sk} = \Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rrk}, \sigma) + \mathsf{sk},$$

where $\mathsf{srk} = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and $\mathsf{rrk} = \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \rho)$.

The one-time public and secret keys are then:

$$\mathsf{opk} = g^k \cdot \mathsf{pk}, \quad \mathsf{osk} = k + \mathsf{sk}.$$

Since $(\mathsf{opk}, \mathsf{osk})$ is a valid key pair in Π_{SIG}, and Π_{SIG} is correct, any signature $\tau = \Pi_{\mathsf{SIG}}.\mathsf{Sign}(\mathsf{osk}, m)$ will verify:

$$\Pi_{\mathsf{SIG}}.\mathsf{Vrfy}(\mathsf{opk}, m, \tau) = 1,$$

for all messages $m \in \mathcal{M}$. Thus, the IB-MSS scheme is correct.

Unlinkability. Assume, for contradiction, that there exists a PPT adversary A that can distinguish between a key pair generated by the IB-MSS scheme and a

random key pair with non-negligible advantage ϵ. We subdivide the proof into two cases: when the adversary A is valid for unlinkability w.r.t. the attributes, and when A is valid for unlinkability w.r.t. the public key.
Unlinkability w.r.t attributes We construct an adversary $\mathsf{A}_{\mathsf{NIKD}}$ that breaks the IND-security of $\mathit{\Pi}_{\mathsf{NIKD}}$:

1. **Setup:**
   - $\mathsf{A}_{\mathsf{NIKD}}$ receives mpk from the IND-security challenger.
   - Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^{\lambda})$.
   - Forwards mpk to A.
2. **Oracle Simulation:**
   - **Key Generation (SKGen, RKGen):** Use the extraction oracle to obtain and return keys.
   - **One-Time Key Generation ($\mathsf{OSKGen}\mathcal{O}$):**
     - Upon receiving a tuple $(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)$ from A, query the $\mathsf{Reveal}\mathcal{O}$ oracle with input (σ, ρ) to receive k. If $\mathsf{opk} \neq g^k \cdot \mathsf{pk}$, set $\mathsf{osk} = \perp$, else set $\mathsf{osk} := k + \mathsf{sk}$.
     - If $\mathsf{flag} = \mathsf{true}$, return osk to A. Else, return 1 to A.
3. **Challenge Phase:**
   - Adversary A outputs $(\sigma^*, \rho^*, \mathsf{st})$.
   - $\mathsf{A}_{\mathsf{NIKD}}$ sends (σ^*, ρ^*) to the IND-security challenger.
   - Receives challenge key k^* (either real or random).
   - Sets $\mathsf{osk}^* = k^*$ and $\mathsf{opk}^* = g^{\mathsf{osk}^* + \mathsf{sk}}$.
   - Provides $(\mathsf{opk}_b, \mathsf{osk}_b, \mathsf{st})$ to A, where:
     - If $b = 0$, $(\mathsf{opk}_b, \mathsf{osk}_b) = (\mathsf{opk}^*, \mathsf{osk}^*)$.
     - If $b = 1$, generate $(\mathsf{opk}_b, \mathsf{osk}_b) \leftarrow\!\!\$\ \mathit{\Pi}_{\mathsf{SIG}}.\mathsf{KGen}(1^{\lambda})$.
4. **Finalization:**
   - A outputs a guess b'.
   - $\mathsf{A}_{\mathsf{NIKD}}$ outputs 0 if $b' = 0$ (real shared key), else 1.

If A distinguishes between the cases with advantage ϵ, then $\mathsf{A}_{\mathsf{NIKD}}$ distinguishes whether k^* is real or random with the same advantage, breaking the IND-security of $\mathit{\Pi}_{\mathsf{NIKD}}$.
Unlinkability w.r.t the public key. Let us consider the following hybrids:

$H_0(\lambda)$**:** The original game
$H_1\lambda)$**:** Same as the original, except that $(\mathsf{opk}_1, \mathsf{osk}_1)$ is generated from KGen as well.

It is straightforward to see that, without any knowledge of sk (which cannot be guessed by the adversary for the discrete log), the pair $g^{k+\mathsf{sk}}$ has the same distribution of g^{sk} for any k. Hence, the two hybrids are indistinguishable.

The proof follows directly from the indistinguishability of the two hybrids since now opk_0 and opk_1 are generated independently using the same key generation algorithm with no input.
Unforgeability. Unforgeability of $\mathit{\Pi}_{\mathsf{IBMSS}}$ follows from the IND-security of $\mathit{\Pi}_{\mathsf{NIKD}}$ and the EUF-CMA security of $\mathit{\Pi}_{\mathsf{SIG}}$. We subdivide the proof into two cases: when the adversary A is valid for unforgeability w.r.t. the attributes, and when A is valid for unforgeability w.r.t. the public key.
Unforgeability w.r.t. the attributes. The proof proceeds via a hybrid argument.

$\mathcal{H}_0(\lambda)$ This is the standard unforgeability game for Π_{IBMSS}.

$\mathcal{H}_1(\lambda)$ Identical to $\mathcal{H}_0(\lambda)$, except that for a random query i^* to $\mathsf{OSKGen}\mathcal{O}$, the key pair $(\mathsf{opk}_{i^*}, \mathsf{osk}_{i^*})$ is generated using $\Pi_{\mathsf{SIG}}.\mathsf{KGen}(1^\lambda)$ instead of using $\mathsf{SharedKey}$.

Suppose there exists an adversary A that can distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$ with non-negligible advantage ϵ. We construct an adversary $\mathsf{A}_{\mathsf{NIKD}}$ that breaks the IND-security of Π_{NIKD}.

1. **Initialization:**
   - Receive the master public key mpk from the IND-SK challenger and forward it to A.
   - Initialize $\mathcal{Q}_{\mathsf{Sign}\mathcal{O}} := \emptyset$, $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \emptyset$, set $j := 0$, and initialize $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$.
   - Choose a random index i^* for a future $\mathsf{OSKGen}\mathcal{O}$ query.
2. **Oracle Simulations:**
   a. **Key Generation Oracles (SKGen, RKGen):**
      For identity queries id, forward id to the $\mathsf{Extract}$ oracle and return the private key $\mathsf{rk}_{\mathsf{id}}$ to A.
   b. **One-Time Key Generation Oracle ($\mathsf{OSKGen}\mathcal{O}$):**
      Upon receiving a query $(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)$ from A:
      - Check if an entry $(\mathsf{opk}', \mathsf{flag}', \sigma', \rho')$ exists in $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$ with $\mathsf{opk} = \mathsf{opk}'$, $\sigma' = \sigma$ and $\rho' = \rho$. If it exists and $\mathsf{flag} = \mathsf{true}$, return osk to A if $\mathsf{opk} \neq \mathsf{pk}$. Else, set $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} = \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} \cup \{\mathsf{opk}, \mathsf{osk}, \mathsf{flag}, \sigma, \rho\}$ and return osk to A if $\mathsf{flag} = \mathsf{true}$. Else, return 1 to A.
      - Otherwise, increment $j := j + 1$.
      - **If $j = i^*$ (Challenge Query):**
        - Send (σ, ρ) as challenge identities to the IND-SK challenger.
        - Receive the challenge shared key k.
        - Set $\mathsf{osk} := k + \mathsf{sk}$.
      - **If $j \neq i^*$:**
        - Compute $\mathsf{rk}_\sigma \leftarrow\$ \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \sigma)$.
        - Compute $\mathsf{osk} := \Pi_{\mathsf{NIKD}}.\mathsf{SharedKey}(\mathsf{mpk}, \mathsf{rk}_\sigma, \rho) + \mathsf{sk}$.
        - Set $\mathsf{opk} := g^{\mathsf{osk}' + \mathsf{sk}}$.
      - $g^{\mathsf{osk} + \mathsf{sk}} \neq \mathsf{opk}$ output $\perp$. Else, add $(\mathsf{opk}, \mathsf{osk}, \mathsf{flag}, \sigma, \rho)$ to $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$ and if $\mathsf{flag} = \mathsf{true}$ output osk to A. Otherwise, output 1 to A.
   c. **Signing Oracle ($\mathsf{Sign}\mathcal{O}$):**
      Upon receiving a query (i, m) from A:
      - Retrieve $(\mathsf{opk}_i, \mathsf{osk}_i)$ corresponding to the i-th query in $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.
      - **If $i \neq i^*$:**
        - Compute $\tau := \Pi_{\mathsf{SIG}}.\mathsf{Sign}(\mathsf{osk}_i, m)$ and return τ to A.
      - **If $i = i^*$:**
        - Since A is valid, it does not query $\mathsf{Sign}\mathcal{O}$ on i^*.
3. **Adversary's Forgery:**
   If A outputs a valid forgery (i^*, m^*, τ^*):
   - Verify if $\Pi_{\mathsf{SIG}}.\mathsf{Vrfy}(\mathsf{opk}_{i^*}, m^*, \tau^*) = 1$.
   - **If valid:** Output $b' = 0$ (guessing k is real).

- **Else:** Output $b' = 1$ (guessing k is random).

If k is real (as in $\mathcal{H}_0$), A can forge with probability ϵ, so $\mathsf{A}_{\mathsf{NIKD}}$ outputs $b' = 0$ with probability related to ϵ.

If k is random (as in $\mathcal{H}_1$), forging under opk' requires breaking the EUF-CMA security of Π_{SIG}, which is negligible. - Therefore, $\mathsf{A}_{\mathsf{NIKD}}$ distinguishes between the two cases, breaking the IND-security of Π_{NIKD}.

Assuming A cannot distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$, any forgery under $\mathcal{H}_1$ can be used to break the EUF-CMA security of Π_{SIG}.

1. **Constructing $\mathsf{A}_{\mathsf{SIG}}$:**
   - Receives opk^* from the EUF-CMA challenger.
   - Generates $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \Pi_{\mathsf{NIKD}}.\mathsf{Setup}(1^\lambda)$.
   - Chooses σ^* and ρ^* not queried by A.
   - Sets i^* corresponding to (σ^*, ρ^*) and assigns $\mathsf{opk}_{i^*} := \mathsf{opk}^*$.
   - Simulates oracles as before, ensuring A does not receive signatures under opk_{i^*}.
2. **Adversary's Forgery:**
   - If A outputs a forgery (i^*, m^*, τ^*), $\mathsf{A}_{\mathsf{SIG}}$ outputs (m^*, τ^*) to the EUF-CMA challenger.

If A can forge in $\mathcal{H}_1$, $\mathsf{A}_{\mathsf{SIG}}$ breaks the EUF-CMA security of Π_{SIG}.
Unforgeability w.r.t. the public key. Suppose there exists an adversary A that can break Unforgeability w.r.t. the public key with non-negligible advantage ϵ. We construct an adversary $\mathsf{A}_{\mathsf{SIG}}$ that breaks the EUF-CMA of Π_{SIG}. For the aim of simplicity, we prove unforgeability assuming that Π_{SIG} is a Schnorr signature.

Therefore, the Unforgeability of $\Pi_{\text{IB-MSS}}$ holds under EUF-CMA security of the Schnorr signature scheme.

1. **Initialization:**
   - Generate the $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\$ \mathsf{MKGen}(1^\lambda)$
   - Initialize $\mathcal{Q}_{\mathsf{Sign}\mathcal{O}} := \emptyset$, $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \emptyset$, set $j := 0$.
   - Receive pk from $\mathsf{A}_{\mathsf{SIG}}$ and forward it to A.
   - Choose a random index i^* for a future $\mathsf{OSKGen}\mathcal{O}$ query.
2. **Oracle Simulations:**
   a. **Key Generation Oracles (SKGen, RKGen):** Generate the sender and retrieval keys using SKGen and RKGen.
   b. **One-Time Key Generation Oracle ($\mathsf{OSKGen}\mathcal{O}$):** Upon receiving a query $(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)$ from A:
      - If an $(\mathsf{opk}', \mathsf{flag}', \sigma', \rho')$ does not exists in $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$ with $\mathsf{opk} = \mathsf{opk}'$, $\sigma' = \sigma$ and $\rho' = \rho$, set $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} := \mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}} \cup \{(\mathsf{opk}, \mathsf{flag}, \sigma, \rho)\}$.
      - Otherwise, increment $j := j + 1$.
      - Return 1 to A. For the validity of A no queries with $\mathsf{flag} = \mathsf{true}$ are admitted.
   c. **Signing Oracle ($\mathsf{Sign}\mathcal{O}$):**
      Upon receiving a query (i, m) from A:

- Retrieve the tuple $(\mathsf{opk}, \sigma, \rho)$ corresponding to the i-th query in $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.
- **If** $i \neq i^*$**:**
  - Compute $\mathsf{rk}_\sigma := \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and $k := \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \mathsf{rk}_\sigma, \rho)$.
  - Query the signing oracle of Π_{SIG} with m and receive $\tau = (R, s)$ back. Compute $\tau' := \tau + H_1(\chi(R), m) \cdot k$ and return τ' to A.
- **If** $i = i^*$**:**
  - Since A is valid, it does not query $\mathsf{Sign}\mathcal{O}$ on i^*.

d. **Adversary's Forgery:**
If A outputs a valid forgery (i^*, m^*, τ^*):
- Compute $\mathsf{rk}_\sigma := \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and $k := \Pi_{\mathsf{NIKD}}.\mathsf{Extract}(\mathsf{msk}, \mathsf{rk}_\sigma, \rho)$.
- Retrieve the tuple $(\mathsf{opk}, \sigma, \rho)$ corresponding to the i-th query in $\mathcal{Q}_{\mathsf{OSKGen}\mathcal{O}}$.
- Parse $\tau^* = (R, s)$ and compute $\tau := \tau^* - H(\chi(R), m) \cdot k$.
- Send τ to $\mathsf{A}_{\mathsf{SIG}}$ and output what $\mathsf{A}_{\mathsf{SIG}}$ outputs.

It is clear from the malleability property of Schnorr signatures that by manipulating the signature received from the signing oracle or the challenge signature received by the adversary, it is always possible to produce a signature that verifies under the challenge public key pk. Hence, if A can forge a signature in the unforgeability game, it breaks EUF-CMA security of the underlying Schnorr signature scheme.

B Implementation Details

Our IB-MSS Instantiation. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of prime order p and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear pairing.

Model the hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \mathbb{Z}_p^*$ as random oracles. Let $\chi : \mathbb{G}_1^* \longrightarrow \{0,1\}^*$ be an injective, efficiently computable encoding that maps a group element to its canonical binary representation (e.g., compressed-point encoding for elliptic-curve groups). The message space is $\{0,1\}^*$.

$\mathsf{MKGen}(1^\lambda)$: Sample $\mathsf{msk} \leftarrow \{0,1\}^\lambda$, choose generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and set $\mathsf{mpk} = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, g_1^{\mathsf{msk}})$. Output $(\mathsf{mpk}, \mathsf{msk})$.

$\mathsf{KGen}(1^\lambda)$: Sample $\mathsf{sk} \leftarrow \mathbb{Z}_p$ and output $(\mathsf{pk} = g_1^{\mathsf{sk}}, \mathsf{sk})$.

$\mathsf{SKGen}(\mathsf{msk}, \mathsf{id})$ and $\mathsf{RKGen}(\mathsf{msk}, \mathsf{id})$: Return $H_1(\mathsf{id})^{\mathsf{msk}}$.

$\mathsf{OPKGen}(\mathsf{srk}, \mathsf{pk}, \rho)$: Compute $k := H_2\big(e(\mathsf{srk}, H_1(\rho))\big)$ and output $\mathsf{opk} := g_1^k\,\mathsf{pk}$.

$\mathsf{OSKGen}(\mathsf{opk}, \mathsf{rrk}, \mathsf{sk}, \sigma)$: Compute $k := H_2\big(e(H_1(\sigma), \mathsf{rrk})\big)$. If $g_1^{k+\mathsf{sk}} = \mathsf{opk}$ output $\mathsf{osk} := k + \mathsf{sk}$, else output $\perp$.

$\mathsf{Sign}(\mathsf{osk}, m)$: Sample $r \leftarrow \mathbb{Z}_p$, let $R := g_1^r$ and $c := H_1\big(\chi(R), m\big)$, set $s := r - c\,\mathsf{osk} \bmod p$, and output $\tau := (R, s)$.

$\mathsf{Vrfy}(\mathsf{opk}, m, \tau)$: Parse $\tau = (R, s)$, compute $c := H_1\big(\chi(R), m\big)$, and accept iff $R = g_1^s\,\mathsf{opk}^{\,c}$.

Parsing Time Costs A crucial task for the recipient in our system is detecting which transaction was sent to them. The recipient must parse through all transactions, compute the stealth address using their secret key and the nonce extracted from the transaction, and verify whether it matches one of the output addresses. On average, this process requires 3.07 ms per transaction, based on 100 trials. This time consists of 0.31 ms to retrieve the transaction, 2.42 ms to compute the shared secret, and 0.34 ms to derive and check the stealth address. While this time is short for individual transactions, parsing a large volume of transactions leads to a substantial increase in time.

To estimate the parsing time on the Bitcoin network, we analyzed confirmed transactions from November 16, 2022, to November 16, 2023. As shown in Fig. 2, the 75th percentile for confirmed transactions per day, week, and month are 462,392, 3,261,635, and 14,132,707, respectively. Based on these figures, the estimated parsing times would be 23.64 minutes for a day, 2.77 hours for a week, and 12.04 hours for a month.

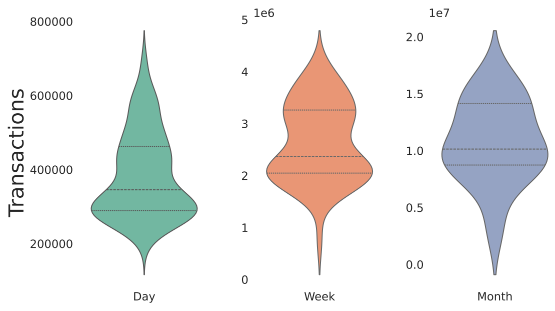

Fig. 2. Violin plot depicting the daily, weekly, and monthly confirmed Bitcoin transactions between November 16, 2022, and November 16, 2023.

One method to reduce parsing time is the use of view tags, as in Monero [32]. A view tag is a small identifier derived from the shared secret and embedded in the transaction data. This allows the recipient to quickly determine whether a transaction belongs to them without performing the full stealth address computation. This technique can reduce parsing time by 40% [22]. However, in our protocol, including view tags would reveal part of the shared secret, compromising the stealthiness of the transaction, which is a key feature of our design.

Another approach to reduce the recipient's computational load is to use the DKSAP protocol. By sharing a secret scanning key with a server, the recipient can delegate transaction monitoring. The server continuously scans the blockchain for relevant transactions, which significantly reduces the computation required on the recipient's side.

In some cases, the sender can also provide specific details, such as the transaction ID or date. This narrows down the range of transactions the recipient needs to check, making the parsing process faster by limiting the search to a smaller set of transactions.