OXFORD

The Computer Journal, 2023, 1–12
https://doi.org/10.1093/comjnl/bxad056
Original Article

Stealth Address Schemes With Fast Retrievability Based On Subgroup Membership Assumptions Related To Factoring

Xin Wang*, Li Lin and Yao Wang

Ant Group, Beijing, China
*Corresponding author: wx352699@antgroup.com

Abstract

Stealth address is a known technique to ensure the privacy (anonymity) of a recipient participating in a certain transaction in a distributed blockchain scenario. However, most existing stealth address schemes require linear judge time and search time $\mathcal{O}(n)$, where n is the number of transactions of a certain block, so the only way to claim transactions for a recipient is to traverse the transaction list to find out whether an ever-arrived transaction belongs to him. To overcome this drawback, we proposed the notion of Fast Stealth Address (FSA), a novel approach that simultaneously preserves privacy and improves search efficiency of recipients. We give a generic construction of FSA scheme under subgroup membership assumption related to factoring and instantiate concrete schemes based on specific number-theoretic assumptions. Our framework mainly improves on two aspects: (i) allowing constant recognize time $\mathcal{O}(1)$ to judge whether a certain block contains recipient's transactions and (ii) allowing logarithmic search time $\mathcal{O}(\log n)$ to find out the precise transactions intended for a recipient. We formalize the security model of an FSA scheme and provide provable security analysis to ensure the security of our constructions. Besides, we implement our schemes to measure their real-world performance on several metrics and give comparison results to stealth address scheme utilized by Monero.

Keywords: Blockchain, Stealth Address, Privacy, Anonymity, Search

1. INTRODUCTION

Blockchain [1] is regarded as a distributed public ledger, which means that all nodes in the network can access the contents recorded in the ledger. As the ledger is completely public, the blockchain itself has no privacy protection, and additional technologies are needed to protect the privacy of transactions.

During the past few years, quite a few cryptographic techniques such as ring signature [2], stealth address [3] and zero-knowledge proof [4] have been employed to ensure transaction privacy for senders, recipients and transaction amount in blockchains. This work focuses on stealth address, a privacy protection technique for recipients of cryptocurrency transactions. Generally speaking, the stealth address scheme allows the sender to create a random one-time address from the actual address of the recipient so that different payments for the same payee are unlinkable. The security requirement of a stealth address scheme can be described as that any external observer cannot distinguish the target recipient from a non-target recipient after observing the corresponding stealth address field. The recipient keeps the actual address key pair to accomplish certain computations so as to find out his transactions from the blockchain ledger. However, most existing privacy-centric schemes, such as Monero [5] and MimbleWimble [6], require the recipient to continuously calculate a value from the claimed one-time address and find the corresponding match in the blockchain. The judge and search time of all existing secure stealth address schemes are thus linear with the total number of all stealth address transactions. This makes it very difficult to retrieve from large-scale transactions, which are often encountered in most practical scenarios. Therefore, it is expected that there will be an improved scheme that can support asymptotically faster retrieval of batch transactions based on stealth addresses, so as to improve the overall performance of the blockchain system under privacy protection.

Motivated by the challenging problem of high-performance private cryptocurrency, recent years have witnessed an extensive literature for privacy-preserving applications where recipient anonymity needs to be protected. In a recent work Beck *et al.* [7] introduced a novel cryptographic primitive named Fuzzy Message Detection (FMD) that allows a server to perform outsourced detection of messages for each recipient. For each recipient, the server will detect ciphertexts and maintain a list of ciphertexts that could be intended for him. This list includes a certain fraction p of false positive. This approach provides the recipients with a tunable parameter p and can therefore trade-off privacy for efficiency. Specifically, the work done by recipient is $O(p \cdot N)$, where N is the total number of ciphertexts. A major drawback is that if full privacy ($p = 1$) is required, the recipient still needs to do $O(N)$ work. Here full privacy means that a ciphertext can be associated with every recipient with the same probability. This encounters the same linear scanning barrier as mentioned before in the cryptocurrency setting. Another recent work [8] introduced the problem of private signaling that abstracts several real-world recipient-anonymous applications including stealth address payments and gave concrete solutions, but the proposed scheme

Received: September 30, 2022. **Revised:** March 07, 2023. **Accepted:** April 26, 2023

must resort to the trusted execution environment (TEE) and thus is not a pure cryptographic solution.

Based on these results, we found that a full privacy scheme with recipient computation efficiency still remains unsolved. So a natural question one could come up with is that whether there is a cryptographic solution for the stealth address payments that achieves full anonymity with lower complexity for the recipient?

Our Contributions. In this work, we answer the above question affirmatively and aim at providing a better framework of stealth address which offers a mechanism that the one-time addresses can be structurally bundled up so that a faster scanning strategy can be applied. For this purpose, we assume there are special nodes called the Helper, which has sufficient storage capacity dedicated to filtering transactions for recipients. The Helper will collect a bunch of stealth addresses at predefined time intervals, e.g. a block generating time, and build a binary search tree to record the addresses in the leaf nodes that are linked to the underlying transactions. From this tree, every recipient could privately judge whether or not the current block contains a transaction for him only by checking the root node, which takes only constant time $\mathcal{O}(1)$ independent of the number of transactions in a certain block. And he can further locate transactions intended for him along paths from the root to all possible leaf nodes in logarithmic complexity $\mathcal{O}(\log n)$, where n is the total number of transactions in a block.

To accomplish the above framework, we propose the notion of Fast Stealth Address (FSA) supporting fast retrieval of cryptocurrency transactions. Specifically, we give a generic construction of FSA scheme based on subgroup membership assumptions related to factoring that naturally adapts to the generation of a binary search tree. In particular, our contributions are 3-fold:

- Formalization of FSA scheme. We first formalize the syntax and the security models for an FSA scheme, capturing the functionality and security (privacy of recipients) requirements that the cryptocurrency practice imposes on private transactions.
- A generic construction of FSA scheme and provable security. We propose a general construction of FSA scheme based on a wide class of cryptographic assumptions named *subgroup membership (SM) assumptions related to factoring*. The general scheme supports logarithmic search time over the entire transaction list and we prove its security in the standard model.
- Concrete FSA schemes with provable security. Based on 2^k-QR assumption and p'-subgroup Decision assumption, respectively, which can be seen as concrete instantiations of SM assumption, we construct two concrete FSA schemes satisfying fast retrievability and security requirements.

To show the process of the proposed FSA, we first define four phases: (i) Setup, (ii) Transaction, (iii) Search Tree Generation and (iv) Retrieval. The sequence of the proposed algorithms utilized by the entities (i.e. Sender, Recipient, Helper) during each of these phases is given as a flowchart in Fig. 1. During the Setup phase (steps 1 and 2), a public parameter (pp) and various key-pairs (i.e. (pk_i, sk_i)) are defined. On the other hand, during the Transaction phase (step 3), a sender prepares an address $Addr$ and attaches it to the transaction. And during the Search Tree Generation phase (step 5), a Helper collects a certain amount of incoming transactions from a block and packs them into a binary search tree BT. During the Retrieval phase (step 6), the recipient logs in his account online and requests for reclaiming his assets

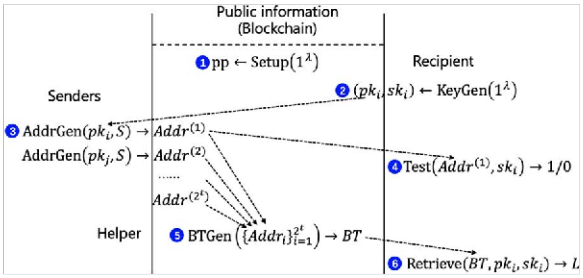

Figure 1. Illustration of FSA flows and algorithms

to the Helper. In response, the Helper finds the results for the available addresses and forwards these results to the Receiver as a transaction list L.

Technical Overview. Our scheme is constructed based on the subgroup membership assumption which is studied in [9–12]. We consider the SM assumption in a slightly different setting that the hardness problem is always related to an RSA modulus $N = pq$ and a trapdoor exists. Recall that the SM assumption posits that a random value of a subgroup $\mathbb{G}_L$ is computationally indistinguishable from a random value of a universe group $\mathbb{G}_U$ such that $\mathbb{G}_L \subseteq \mathbb{G}_U$. In our setting, the membership in the subgroup $\mathbb{G}_L$ is decidable with the help of some trapdoor information τ related to the factorization of N. That is, one can decide if a group element $x \in \mathbb{G}_U$ resides in subset $\mathbb{G}_L$ with τ.

In our scheme, a stealth address is always constructed by the sender as a 'mixed hint' indicating whether or not a recipient will receive a transaction. Specifically, each recipient owns a modulus which defines a subgroup membership problem. When initiating a transaction, the sender chooses a random non-subgroup element denoting 'yes' for the intended recipient and chooses random subgroup elements representing 'no' for all unintended recipients. Then the sender combines all these hints into a mixed hint and uses it as the stealth address. Whenever coming online, the intended recipient could derive from this address a hint 'yes' using his secret key, while all other recipients could obtain a hint 'no'. Besides, any polynomial-time observer cannot tell whether an element implies 'yes' or 'no' relative to a recipient's modulus.

We explain the basic idea of the scheme in more detail. Suppose there are n recipients $\{R_1, \cdots, R_n\}$, and each recipient possesses a key pair (pk_i, sk_i) $(i \in [n])$. A public key pk_i consists of group parameters (N_i, g_i), where g_i is a random generator of a cyclic group $\mathbb{G}_U^{(i)} \subseteq \mathbb{Z}_{N_i}$. The group parameters implicitly define an instance of SM problem $SM_{(\mathbb{G}_U^{(i)}, \mathbb{G}_L^{(i)})}$. When a sender initiates a transaction intended for a target recipient R_i, he chooses group elements $h_1, h_2, \cdots, h_{i-1}, g_i, h_{i+1}, \cdots, h_n$ acting as hints for recipients such that $h_j \xleftarrow{R} \mathbb{G}_L^{(j)}$ for $j \neq i$ and $g_i \in \mathbb{G}_U^{(i)} \backslash \mathbb{G}_L^{(i)}$ (We require that both $\mathbb{G}_U^{(j)}$ and $\mathbb{G}_L^{(j)}$ allow efficiently uniform sampling.). Then the sender computes the 'mixed hint' $H = [h_1, h_2, \cdots, h_{i-1}, g_i, h_{i+1}, \cdots, h_n]$ and packs it into the transaction. The square bracket here is a bi-directional function such that every recipient could recover from H his part of hint. Since each R_j possesses the secret key sk_j, he can determine whether or not his hint lies in $\mathbb{G}_L^{(j)}$ and therefore knows if the corresponding transaction belongs to him. In particular, $g_i \notin \mathbb{G}_L^{(i)}$ so that R_i knows he gets a transaction. The anonymity of R_i maintains since for all $j \neq i$, R_j cannot know which of $\{h_i\}_{i \neq j}$ comes from the non-subgroup since the subgroup membership problem of $\mathbb{G}_L^{(i)}$ in $\mathbb{G}_U^{(i)}$ is intractable without the trapdoor information sk_i.

Note that this achieves the security notion of full anonymity in the sense that for any external observer, the probability that he guesses the correct identity of the target recipient is exactly $1/n$.

Up to now, we have finished the demonstration of the ingredients that a 'mixed hint' contains. It seems that the only way to look for possible transactions is to search one by one. For example, given two values H_1, H_2, each recipient starts from H_1, splits it and judges the membership of individual elements separately and then repeats the same process to H_2 to find out all the potential targets. How can we achieve a better strategy than linear scan?

The picture will be clear if we combine two or more stealth addresses together. Given

$$H_1 = [h_1, h_2, \cdots, h_{i-1}, g_i, h_{i+1}, \cdots, h_n],$$

$$H_2 = [h'_1, h'_2, \cdots, h'_{i-1}, h'_i, h'_{i+1}, \cdots, h'_n],$$

we can generate a combined value

$$H_{(1,2)} = [h_1h'_1, \cdots, h_{i-1}h'_{i-1}, g_ih'_i, h_{i+1}h'_{i+1}, \cdots, h_nh'_n]$$

which can be seen as a component-wise multiplied value. The point is that the membership of group elements in each position preserves if there ever exists a non-subgroup element somewhere in an H-value. As is shown above, the membership of group element in the i-th position of $H_{(1,2)}$ remains the same as H_1, even after the joining of H_2. This will always hold no matter how many H-values participate in the generation of the combined value. From this single value, R_i can know if any transactions ever arrive, thus the recognition complexity is $\mathcal{O}(1)$ independent of the total number of transactions. Furthermore, given m stealth addresses, if we build a binary tree tying every two adjacent values, we can achieve a $\log m$ scan over the entire transaction list. The leaf node simply records the H values and each non-leaf node is computed as the multiple of its children. The number of the leaves corresponds to the number of stealth transactions. Whenever a user comes online, he can check whether the root is a non-subgroup value with respect to his modulus. If not, then he did not receive a transaction. If yes, then he can check which of the child nodes is a non-subgroup value with respect to his modulus and recursively can find the corresponding H_j value. This process will be elaborated in Section 4.

Related work. Due to its decentralization, tamper-resistance, openness and transparency, blockchain has gained wide application in the fields of finance, smart grid [13, 14], internet of things [15–17] and supply chains [18, 19]. At the same time, openness and transparency inevitably threaten the privacy of users in transactions.

A significant amount of work has been done in privacy-preserving blockchain scenarios, such as FMD [7], private signaling [8], stealth address etc.

The initial stealth address scheme was proposed by a Bitcoin Forum member known as 'ByteCoin' in 2011, which was then improved in [5] by introducing the random ephemeral key pair and fixing the issue that the sender might change the mind and reverse the payment. Later on, a dual-key enhancement to the previous stealth address schemes was implemented in 2014, which utilized two pairs of cryptographic keys for designated third parties (e.g. auditors, proxy servers, read-only wallets, etc.) removing the unlinkability of the stealth addresses without simultaneously allowing payments to be spent. Stealth address is considered as a lightweight solution to enhance privacy and has been widely deployed by cryptocurrency communities, e.g. Bitcoin Hierarchical Deterministic Wallets (HD Wallets) [20], Cryptonote [21], Monero [5, 22], Zcash [4] and other blockchain technologies.

The rest of the paper is organized as follows. Section 2 gives the mathematical background knowledge and definition of subgroup membership assumption related to factoring. Section 3 gives a brief overview of the system model applied by stealth address, followed by the formal definition of newly proposed FSA scheme. In Section 4, we present a generic construction of FSA scheme and analyze its security. In Section 5, we give concrete instantiations under 2^k-QR and p'-subgroup decision assumptions, followed by security analysis. In Section 6, we give implementation and evaluation results of our concrete schemes. Finally, Section 7 concludes this work and gives future direction in this line of research.

2. BACKGROUND

2.1. Notations

$\lambda \in \mathbb{N}$ denotes the security parameter and 1^λ denotes its unary form. If x is a binary string, then $|x|$ denotes its bit-length. If S is a finite set then $|S|$ denotes its size. We use $x \xleftarrow{R} S$ to represent sampling an element x uniformly from set S. For an integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. Algorithms are randomized unless explicitly noted. PPT stands for probabilistic polynomial time. A function is *negligible* if for every $c > 0$ there exists an x_0 such that for all $x > x_0$, $f(x) < 1/x^c$.

2.2. Blockchain

Blockchain [23] is a distributed ledger where data are stored in chained blocks publicly accessible to the network nodes. Any activity or exchange of resources made by network participants is stored in the blockchain as a transaction. Transactions are grouped and inserted into a block. Each block contains the hash of the previous block that creates a link between the blocks, comparable with a chain, making the blocks immutable. Before transactions are entered into the blockchain, peers of the network must agree on their validity. In distributed computing, this problem is known as consensus [24]. According to the way the network nodes are selected, a blockchain can be classified as permissionless or permissioned. A permissionless blockchain permits to anonymous nodes to participate in consensus. In contrast, in a permissioned blockchain, only selected nodes are authorized to join the network and participate in distributed consensus. In this paper, we consider a permissioned blockchain.

Before presenting our result, we review the definition of composite order groups and describe the subgroup membership assumption on which the security of our stealth address schemes are based.

2.3. Quadratic Residues and Jacobi Symbols

Let $N = pq$ be the product of two odd primes p and q. We use $\mathbb{J}(N)$ to denote the set of elements in $\mathbb{Z}_N^*$ with Jacobi Symbol 1; we use $\mathbb{QR}(N)$ to denote the set of quadratic residues (squares) modulo N. The groups $\mathbb{J}(N)$ and $\mathbb{QR}(N)$ have orders $\frac{\phi(N)}{2}$ and $\frac{\phi(N)}{4}$, respectively.

2.4. r-th residues and residue symbol

For each integer $r \geq 2$, the subgroup of r-th power residues [25, 26] modulo N is denoted by $(\mathbb{Z}_N^*)^r = \{x^r \mid x \in \mathbb{Z}_N^*\}$. We say that $y \in \mathbb{Z}_N^*$ is an r-th residue modulo N if there is an $x \in \mathbb{Z}_N^*$ such that $y = x^r \bmod N$. If $y = x^r$ has no solution in $\mathbb{Z}_N^*$, then y is called an r-th non-residue.

Definition 2.1. (r-th residue symbol). Let p be an odd prime and let $r \geq 2$ such that $r \mid p-1$. Then the symbol

$$\left(\frac{a}{p}\right)_r = a^{\frac{p-1}{r}} \mod p$$

is called the r-th residue symbol modulo p.

For any integer a with $\gcd(a,p)=1$, a is an r-th residue modulo p if and only if

$$\left(\frac{a}{p}\right)_r = 1 \mod p$$

2.5. Chinese Remainder Theorem

Theorem 2.1. Let the integers $n_1, n_2, \cdots, n_k$ be positive which are relatively prime in pair, i.e. $\gcd(n_i, n_j) = 1$ when $i \neq j$. Furthermore, let $n = n_1 n_2 \cdots n_k$ and let $x_1, \cdots, x_k$ be integers. Then the system of congruences

$$\begin{aligned} x &\equiv x_1 \mod n_1 \\ x &\equiv x_2 \mod n_2 \\ &\vdots \\ x &\equiv x_k \mod n_k \end{aligned}$$

has a simultaneous solution x to all of the congruences, particularly there exists exactly one solution $0 \leq x < n$.

The unique solution x of the simultaneous congruences satisfying $0 \leq x < n$ can be calculated as

$$x = \left(\sum_{i=1}^{k} x_i r_i s_i\right) \mod n,$$

where $r_i = n/n_i$ and $s_i = r_i^{-1} \mod n_i$ for $i \in [k]$. As a consequence of the Chinese Remainder Theorem (CRT), any positive integer $a < n$ can be uniquely represented as a k-tuple $[a_1, a_2, \cdots, a_k]$ and vice versa, where a_i denotes the residue $a_i \mod n_i$ for each $i \in [k]$.

2.6. Subgroup Membership Assumption Related to Factoring

We will let N be a product of two different primes throughout the remaining parts of the paper.

Let a 'universe' group $\mathbb{G}_U \subseteq \mathbb{Z}_N^*$ and a 'language' group $\mathbb{G}_L \subseteq \mathbb{Z}_N^*$ be multiplicative subgroups of $\mathbb{Z}_N^*$ such that $\mathbb{G}_L \leq \mathbb{G}_U$. The subgroup membership problem denoted by $SM_{(\mathbb{G}_U,\mathbb{G}_L)}$ is to decide whether or not a given element $x \xleftarrow{R} \mathbb{G}_U$ belongs to $\mathbb{G}_L$. More formally,

Definition 2.2. The subgroup membership problem is always associated with a pair of algorithms (Gen, Decide) with the following properties. There is a PPT group generator algorithm Gen that takes as input a security parameter 1^λ and outputs a tuple $I_{\mathbb{G}_U} = ([\mathbb{G}_U], [\mathbb{G}_L], g, h, \tau)$, where $[\mathbb{G}_U]$ (resp. $[\mathbb{G}_L]$) is the description of a universe group $\mathbb{G}_U$(resp. $[\mathbb{G}_L]$), g (resp. h) is the generator of $\mathbb{G}_U$ (resp. $\mathbb{G}_L$) and τ is the trapdoor such that:

- $SM_{(\mathbb{G}_U,\mathbb{G}_L)}$ is computationally intractable without the trapdoor. Formally, for any PPT adversary $\mathcal{A}$, the subgroup distinguishing advantage of $\mathcal{A}$

$$Adv_{\mathcal{A}}^{SM_{(\mathbb{G}_U,\mathbb{G}_L)}} = \Big|\Pr[\mathcal{A}([\mathbb{G}_U],[\mathbb{G}_L],x)=1 \mid x \xleftarrow{R} \mathbb{G}_U] - \Pr[\mathcal{A}([\mathbb{G}_U],[\mathbb{G}_L],x)=1 \mid x \xleftarrow{R} \mathbb{G}_L]\Big|$$

must be negligible in λ.
- $\mathsf{Decide}(x,\tau) \rightarrow \{0,1\}$ is a deterministic polynomial-time algorithm that takes as input a group element $x \in \mathbb{G}_U$ and the trapdoor τ, outputs 1 if x lies in the language group $\mathbb{G}_L$ or 0 otherwise.

The following lemma is basic but useful in constructing our scheme.

Lemma 2.1. Let $\mathbb{G}_U$ be a group, and $\mathbb{G}_L$ be a subgroup of $\mathbb{G}_U$. For any $g \in \mathbb{G}_U$, and $h \in \mathbb{G}_L$, $g \cdot h \in \mathbb{G}_L$ if and only if $g \in \mathbb{G}_L$.

2.7. Instantiations

Next we instantiate the SM assumption based on the 2^k-QR [27] and p'-subgroup decision assumptions.

2.7.1. *Instantiation Under the 2^k-QR Assumption*

We first recall the 2^k-QR assumption proposed by Joye *et al.* [27] (called Gap 2^k-Residuosity Assumption in the context of [27]) in its original form and further analyze how to transform it to the standard form according to our definition of SM.

Definition 2.3. (2^k-QR Assumption). Let Gen be a PPT algorithm which, given a security parameter λ, outputs primes p and q such that $p, q \equiv 1 \mod 2^k$. The 2^k-QR problem in $\mathbb{Z}_N^*$ consists in distinguishing a uniform element of V_0 from a uniform element of V_1 given only $N = pq$, where V_0 and V_1 are defined as follows:

$$V_0 = \{x \mid x \in \mathbb{J}(N) \backslash \mathbb{QR}(N)\},$$

$$V_1 = \{y^{2^k} \mod N \mid y \in \mathbb{Z}_N^*\}.$$

The 2^k-QR assumption posits that the advantage $Adv_{\mathcal{A}}^{2^k\text{-QR}}(\lambda)$, defined as

$$Adv_{\mathcal{A}}^{2^k\text{-QR}} = \Big|\Pr[\mathcal{A}(N,k,x)=1 \mid x \xleftarrow{R} V_0] - \Pr[\mathcal{A}(N,k,x)=1 \mid x \xleftarrow{R} V_1]\Big|$$

is negligible for any PPT distinguisher $\mathcal{A}$.

We slightly change the form of primes p and q such that $p = 2^k p' + 1, q = 2^k q' + 1$, where p', q' are also primes. The benefit to do so is that both V_0 and V_1 become multiplicative (cyclic) groups. To see so, we follow the discussion in [28]. In this setting, we can decompose $\mathbb{Z}_N^*$ as an internal direct product

$$\mathbb{Z}_N^* \cong G_{p'q'} \cdot G_{2^k} \cdot T_{2^k},$$

where each group G_t is a group of order t. As already analyzed in [28], a random element $g \in \mathbb{J}(N)\backslash\mathbb{QR}(N)$ will have order $2^k p'q'$ with overwhelming probability. Let $\mathbb{G}_U = G_{p'q'} \cdot G_{2^k}$ and $\mathbb{G}_L = G_{p'q'}$. For now the 2^k-QR assumption can be redeemed as that it is computationally infeasible to distinguish elements of $\mathbb{G}_U\backslash\mathbb{G}_L$ from that of $\mathbb{G}_L$. Besides, we can efficiently determine the membership of x in $\mathbb{G}_L$ provided that the trapdoor (p, q) is available. The algorithm Decide is defined as: on input x, (p, q), outputs 1 if $\left(\frac{x}{p}\right)_{2^k} = 1$ and $\left(\frac{x}{q}\right)_{2^k} = 1$, outputs 0 otherwise.

2.7.2. *Instantiation Under the p′-subgroup Decision Assumption with Any Strong RSA Integer*

Consider now a strong RSA modulus $N = pq$, where $p = 2p' + 1, q = 2q' + 1$, and p', q' are also primes. Let $N' = p'q'$. Consider the group $\mathbb{Z}_N^*$ and the subgroup of $\mathbb{Z}_N^*$. We can decompose $\mathbb{Z}_N^*$ as an internal direct product

$$\mathbb{Z}_N^* = G_{N'} \cdot G_2 \cdot T = G_{p'} \cdot G_{q'} \cdot G_2 \cdot T,$$

where each group G_t is a cyclic group of order t, and T is the subgroup generated by $(-1 \mod N)$.

This decomposition is unique, except the choice of G_2. For any $x \in \mathbb{Z}_N^*$, we can express x uniquely as $x = x(G_{p'})x(G_{q'})x(G_2)x(T)$, where for each G_t, $x(G_t) \in G_t$ and $x(T) \in T$. By setting $\mathbb{G}_U = G_{N'}$ and $\mathbb{G}_L = G_{p'}$, the p'-subgroup Decision assumption is that for a properly generated N, the distribution $\{x \mid x \leftarrow G_{N'}\}$ and $\{x \mid x \leftarrow G_{p'}\}$ are computationally indistinguishable. We can efficiently determine the membership of x in $\mathbb{G}_L$ provided that the trapdoor p' is available. The algorithm Decide is defined as: on input x, p', outputs 1 if $x^{p'} \mod N = 1$, outputs 0 otherwise.

3. FSA SCHEME

In this section, we present the system model adopted by an FSA scheme. Then we formalize the definition and security model for FSA scheme.

First, we emphasize that the principal motivation of the proposed FSA is to overcome the limitation in the existing SA schemes that allow an exhausting search and thus become inefficient when the transaction scale increases. Thus, FSA is aimed at supporting an SA scheme with the novel provision for fast retrieval of the oriented transactions and thereby avoiding linear lookups.

3.1. System Model

We assume the standard blockchain communication model where users communicate with each other over a partially synchronous network. We consider a permission-based model, where explicit registration is required for becoming a member of the system. Users can have the following roles: Senders, and Recipients, who utilize the service provided by the blockchain. They submit requests in the form of transactions. There are also Helpers which are special blockchain nodes with sufficient storage capacity. Their task is to collect the stealth addresses of transactions, then build auxiliary search trees and help recipients retrieve matching stealth addresses from the tree.

As shown in Fig. 2, a transaction sender generates a stealth address *Addr* for a recipient and appends it to the fragment 'receiver' of a transaction *Tx*. After a period of time, a bunch of transactions have been encapsulated into a block on the blockchain. Later, a helper on chain utilizes the information of the entire list of transactions of a block to build a complete binary search tree to help recipients with fast searching. If a recipient comes online and wants to decide if a block contains his transactions, he first downloads the root node of this block from the helper. Using secret key sk_i, he is then able to decide the existence of his transactions in constant time. If transactions exist, he can further interact with the helper layer by layer and finally find out the exact positions of his transactions in the block.

3.2. Threat Model

We assume that senders are honest and rational which means that before the transaction creation, they will compute the address value correctly. Besides, helpers are honest in the sense that they will collect addresses honestly and insert leaves in the tree correctly.

We assume that an adversary has the capabilities to perform the following attacks: (1) The external eavesdropper, as an adversary can perform passive attack to deduce the target recipients from the available lists of stealth addresses. (2) The internal malicious user, as an adversary can perform collusion attack to threaten the privacy of an honest recipient. With this kind of adversary, we define anonymity for the proposed stealth address scheme based on the security game described in Section 3.4.

Note that the adversary in the above model is allowed to generate key pairs and corrupt some recipients to obtain the corresponding secret key of its choice. This captures the security requirement that any curious user could observe the incoming transactions and the derived addresses to analyze the transaction flow, whether it be a newly joined user or a corrupted user.

3.3. Definition of FSA Scheme

Intuitively, a stealth address scheme is analogous to public key encryption with some modifications. Key generation is the same, equipping a user i with a pair of keys (pk_i, sk_i). The encryption procedure is substituted by a randomized stealth address generation algorithm AddrGen that on input solely a public key without a payload (message) produces an address value *Addr*. There is no decryption procedure but a match judging algorithm Test that on input an address and secret key sk_i detects whether *Addr* was derived from the corresponding public key pk_i. For privacy, we require *target anonymity*: any polynomial time adversary who is given an honestly generated address and unintended recipients' secret keys must be unable to distinguish between target recipient's address and random values in the range of valid addresses.

Formally, a *Fast Stealth Address* scheme FSA consists of a tuple of PPT algorithms (Setup, KeyGen, AddrGen, Test, BTGen, Retrieve) defined as follows.

- $\text{Setup}(1^\lambda) \rightarrow pp$: On input a security parameter λ in unary, outputs public parameters *pp*. The public parameters *pp* are common inputs used by all users in the system. *pp* define the user identity space *I*, and the maximum size of the anonymity set *S*. We use *pp* as an implicit input to every algorithm.
- $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$: On input a security parameter, outputs a key pair (pk_i, sk_i) for user i, in which pk_i is called user's public key and sk_i is called user's secret key.
- $\text{AddrGen}(pk_i, S) \rightarrow Addr$: On input a target user's public key pk_i, and a list of public keys $S = \{pk_1, \cdots, pk_n\}$, outputs a stealth address *Addr*. We assume that (1) the input public key pk_i and the corresponding secret key sk_i is a valid key pair output by KeyGen and $pk_i \in S$, (2) the list of public key *S* is an ordered

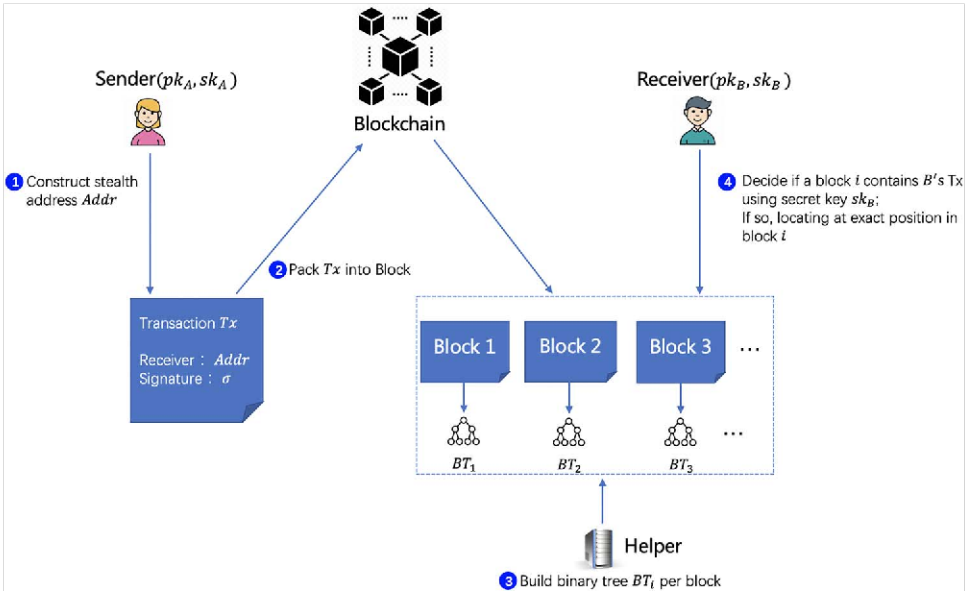

Figure 2. System Model.

set, namely, when it is used in algorithms, the public keys are ordered and each one has an index.

- $\mathsf{Test}(Addr, sk_i) \to \{0,1\}$: On input a transaction address $Addr$ and user i's secret key sk_i, output a bit b representing whether or not the transaction tied to $Addr$ belongs to user i.
- $\mathsf{BTGen}(\{Addr^{(i)}\}_{i=1}^{T}) \to BT$: Suppose that a block contains $T = 2^t$ transactions. This algorithm builds a complete binary-tree BT which supports fast retrieval of transactions.
- $\mathsf{Retrieve}(BT, pk_i, sk_i) \to L$: On input user i's key pair, this algorithm searches for all transactions belonged to user i according to BT and outputs a set L containing the transaction identifiers.

We require that FSA satisfies functional and security properties, which can be referred to as correctness, *fast retrievability* and anonymity.

Correctness. FSA is correct if valid matches always meet the Test algorithm. More formally, the following holds:

$$\Pr\left[\mathsf{Test}(Addr, sk_i) \to 1 \;\middle|\; \begin{array}{l}(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda);\\ Addr \leftarrow \mathsf{AddrGen}(pk_i, S)\end{array}\right] = 1$$

Fast Retrievability. FSA satisfies fast retrievability if there exists an algorithm Retrieve that outputs a list of transactions L in time $\mathcal{O}(\ell \log n)$, where n is the total number of transactions and ℓ is the number of transactions belonged to user i.

Note that the first four algorithms constitute a normal stealth address scheme which may not be equipped with a fast retrieving strategy.

3.4. Security Model

A secure stealth address scheme should satisfy anonymity, which guarantees that no one can find a match between a stealth address generated from a target recipient's public key and the recipient's long-term public key without the corresponding secret key. Formally, we define the security properties in the following game.

Anonymity. A (fast) stealth address scheme satisfies anonymity, if for any PPT adversary $\mathcal{A}$, it holds that $\mathcal{A}$ has at most negligible advantage in the following game $\mathsf{Game}_{\mathcal{A}}^{Anon}(\lambda)$ with a challenger $\mathcal{C}$.

- **Setup**: $\mathcal{C}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda; r_i)$ for all $i \in [n]$, where $n = \mathrm{poly}(\lambda)$ and r_i is the randomness used in KeyGen. $\mathcal{C}$ sets $S = \{pk_1, \cdots, pk_n\}$ and initializes an empty set L_c. Finally, $\mathcal{C}$ sends (pp, S) to $\mathcal{A}$.
- **Query Phase**: $\mathcal{A}$ can adaptively query the following oracle:
  - Corrupting oracle $\mathsf{Corrupt}(\cdot)$: On input an index $i \in [n]$ such that $pk_i \in S$, this oracle returns sk_i and adds pk_i to L_c.
- **Challenge**: The previous phase is repeated a polynomial number of times until $\mathcal{A}$ selects a target public key pk_{i^*} such that $pk_{i^*} \in S \wedge pk_{i^*} \notin L_c$ and sends it to $\mathcal{C}$. The challenger chooses a random bit $b \in \{0,1\}$, if $b = 0$ the challenger chooses a random element in the range of a valid address and returns it to $\mathcal{A}$. Otherwise, the challenger computes $Addr^* \leftarrow \mathsf{AddrGen}(pk_{i^*}, S)$ and returns $Addr^*$ to $\mathcal{A}$.
- **Guess**: $\mathcal{A}$ outputs a bit b' as its guess of b. If $b' = b$, $\mathcal{C}$ outputs 1, otherwise 0.

We define the advantage of $\mathcal{A}$ in the above game as

$$Adv_{\mathcal{A}}^{Anon} = \left|Pr[b' = b] - 1/2\right|$$

We say that FSA is anonymous if for any PPT adversary $\mathcal{A}$, the above advantage is negligible.

4. GENERIC CONSTRUCTION OF FSA SCHEME

4.1. Main Construction

In this section, we demonstrate how to construct an FSA scheme from subgroup membership problem.

The generic stealth address scheme for subgroup membership problem $SM_{(\mathbb{G}_U, \mathbb{G}_L)}$ is defined as follows:

- $\mathsf{Setup}(1^\lambda)$: Takes as input a security parameter λ, and chooses an integer n. Output public parameters $pp = n$.
- $\mathsf{KeyGen}(1^\lambda)$: Chooses random primes p_i, q_i according to $I_{\mathbb{G}_U}$ and compute $N_i = p_i q_i$. Choose a random generator g_i of universal

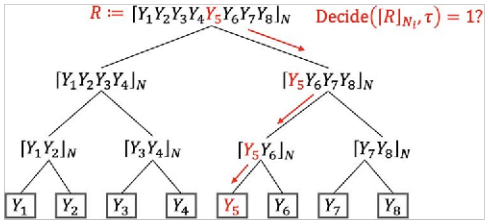

Figure 3. Toy example: Searching from eight transactions requires one computation to identify and three computations to locate.

group $\mathbb{G}_U$ and a random generator h_i of language group $\mathbb{G}_L$. Set $pk_i := (N_i, g_i, h_i), sk_i = p_i$. Output key pair (pk_i, sk_i).

- AddrGen(pk_i, S): To generate a transaction address for user i, first choose a random $r \in \{0,1\}^\lambda$. Using CRT, compute an integer such that

$$\begin{cases} Y \equiv g_i \cdot h_i^r \mod N_i \\ Y \equiv h_j^r \mod N_j, \quad \forall j,\ j \neq i \end{cases}$$

  Set the transaction address as $Addr_{pk_i} = Y$ and pack it into the transaction. Note that the group element with respect to the target user is always a non-language group element according to Lemma 2.1.
- Test$(Addr, pk_i, sk_i)$: Takes as input a transaction address $Addr = Y$, first compute $y_i = Y \mod N_i$. Then run Decide algorithm on input $\tau = sk_i$ and y_i and get the answer b. If $b = 1$, user i rejects the transaction tied to $Addr$; otherwise, accepts and stores the asset into its wallet.
- BTGen$(\{Addr^{(i)}\}_{i=1}^n)$: Suppose that a block contains $n = 2^t$ transactions and set $Y_i := Addr^{(i)}$ (for $i \in [n]$). We denote by $Y_{(i,j,\ldots)}$ the multiplication of integers $Y_i, Y_j, \ldots$, i.e. $Y_{(i,j,\ldots)} = Y_i \cdot Y_j \cdots$. To support fast retrieving within a bunch of transactions, this algorithm builds a complete binary-tree BT following the basic procedure below.

  1) Extract each stealth address from each transaction of a block, which are denoted by $Y_1, Y_2, \cdots, Y_n$.
  2) For $1 \leq i \leq \frac{n}{2}$, compute combined elements $Y_{(2i-1,2i)} = Y_{2i-1}Y_{2i} \mod N$, which form $n/2$ groups of transactions, and store all the middle nodes $Y_{(1,2)}, Y_{(3,4)}, \cdots, Y_{(n-1,n)}$ in order.
  3) Iteratively execute step (2) until all nodes are combined as a single root node. The process can be easily followed as shown in Fig. 4.

- Retrieve(BT, pk_i, sk_i): Takes as input the binary-tree BT and user i's public-secret key pair, search for all transactions belong to user i following the procedure of Algorithm 1.

The functionality being computed by algorithm Retrieve performs the following elementary operations:

1. Begin from the top of the tree by evaluating modular arithmetic on the node values;
2. Proceed to the t-th depth of the tree applying a binary search condition on that the current value is not a subgroup element;
3. Locate the leaf node when the search ends up with the last layer.

An Example. Fig. 3 shows a specific example to provide a better understanding of the fast retrieval process. Assume that a block contains eight transactions. Each transaction associates with an invisible address Y_k $(k = 1, \ldots, 8)$. Suppose that Y_5 is formed

Algorithm 1 Pseudocode of Fast Retrieving Algorithm

Input: i: user i's public key $pk_i = (N_i, g_i, h_i)$, secret key $sk_i = \tau$; t: depth of transaction binary tree; T: total number of transactions;
Output: transaction identifiers that belong to user i;

Set $d = 0$, representing the current search depth of the tree. Pick out the value of root node, namely $Y_{(1,2,\cdots,T)}$, and set $R = Y_{(1,2,\cdots,T)}$. Compute root value modulo user i's moduli: $R_i = R \mod N_i$;
while Decide$(R_i, \tau) \neq 1$ **do**
  for $j = 1; j < t; j{+}{+}$ **do**
    Set $d = d + 1$. Pick out the left child of the current node, namely $Y_{(1,2,\cdots,T/2)}$, set $R = Y_{(1,2,\cdots,T/2)}$;
    judge whether current node is a leaf node (by testing if $d = t$):
    if $d == t$ **then**
      set $R_i = R \mod N_i$
      if Decide$(R_i, \tau) \neq 1$ **then**
        output $Leaf_i$
      else
        output $Leaf_{i+1}$
      end if
    else
      compute $R_i = R \mod N_i$,
      if Decide$(R_i, \tau) \neq 1$ **then**
        Pick out the left child of the current node $Y_{(1,2,\cdots,n/2^{d+1})}$, set $R = Y_{(1,2,\cdots,n/2^{d+1})}$ and $d = d + 1$
      else
        Pick out the left child of the sibling node $Y_{(n/2^d+1,\cdots,3n/2^{d+1})}$, set $R = Y_{(n/2^d+1,\cdots,3n/2^{d+1})}$ and $d = d + 1$.
      end if
    end if
  end for
end while

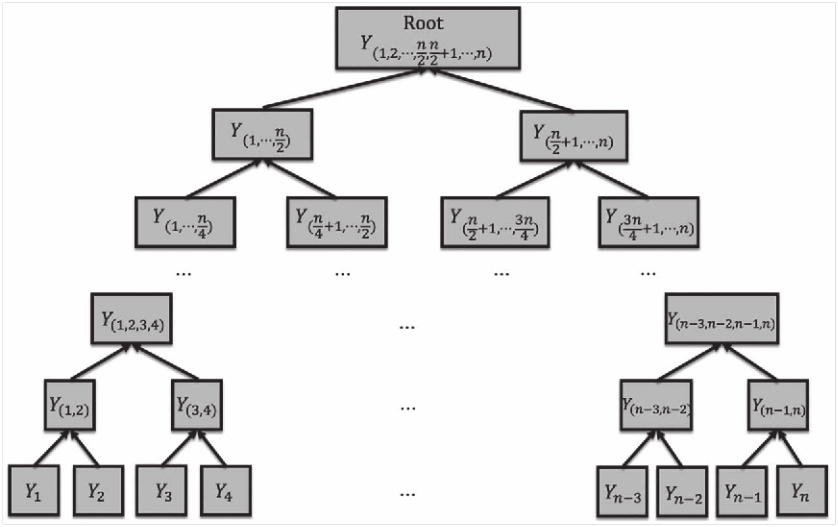

Figure 4. Block Tree Generation.

by a sender targeted at recipient with public key $pk_i = N_i$. For simplicity, we use the notation $\lceil\cdot\rfloor_N$ to stand for an integer value modulo N. Recipient i uses his secret key $sk_i = \tau$ to run algorithm $\mathsf{Decide}(R_i, \tau)$ where $R_i = \lceil R \rfloor_{N_i}$ to judge whether this block contains a target. If so, he does further computations likewise along the path from root to the leaf as indicated by the arrows.

4.2. Security Analysis

We prove the following theorem to illustrate the security of the generic construction of stealth address scheme.

Theorem 4.1. If the $SM_{(\mathbb{G}_U,\mathbb{G}_L)}$ assumption holds in group $\mathbb{Z}_N^*$, then the stealth address scheme satisfies anonymity.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the anonymity of stealth address scheme with a non-negligible advantage ϵ_{Anon}. Then we can construct a PPT simulator $\mathcal{B}$ that solves the $SM_{(\mathbb{G}_U,\mathbb{G}_L)}$ problem.

The input of $\mathcal{B}$ is $(N, [\mathbb{G}_U], [\mathbb{G}_L], x)$ and the target of $\mathcal{B}$ is to distinguish that $x \xleftarrow{R} \mathbb{G}_U$ or $x \xleftarrow{R} \mathbb{G}_L$. $\mathcal{B}$ simulates the $\mathsf{Game}_{\mathcal{A}}^{Anon}$.

- **Setup**: $\mathcal{B}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(pk_i := (N_i, g_i, h_i), sk_i := p_i) \leftarrow \mathsf{KeyGen}(1^\lambda; r_i)$ for all $i \in [n]$, where $n = \mathrm{poly}(\lambda)$ and r_i is the randomness used in KeyGen. $\mathcal{C}$ sets $S = \{pk_1, \cdots, pk_n\}$ and initializes an empty set L_c. Finally, $\mathcal{B}$ sends (pp, S) to $\mathcal{A}$.
- **Query Phase**: $\mathcal{A}$ can adaptively query the following oracle:
  - Corrupting oracle $\mathsf{Corrupt}(\cdot)$: On input an index $i \in [n]$ such that $pk_i \in S$, this oracle returns $sk_i = p_i$ and adds $pk_i = (N_i, g_i, h_i)$ to L_c.
- **Challenge**: $\mathcal{A}$ selects a public key $pk_{i^*} \in S \wedge pk_{i^*} \notin L_c$, where pk_{i^*} is implicitly set by $\mathcal{B}$ as (N, x, h_{i^*}) and sends it to $\mathcal{B}$. $\mathcal{B}$ picks a random $r \in \{0,1\}^\lambda$ and a random bit $b \in \{0,1\}$. With the help of CRT, $\mathcal{B}$ computes an integer Y such that

$$\begin{cases} Y \equiv x^b \cdot h_{i^*}^r \mod N_{i^*} \\ Y \equiv h_j^r \mod N_j, \qquad \forall j,\ j \neq i^* \end{cases}$$

Then $\mathcal{B}$ sets $Addr^* = Y$ and then returns it to $\mathcal{A}$.

- **Guess**: Finally, $\mathcal{A}$ outputs a bit b' as its guess of b. If $b' = b$, $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0.

There are two cases to distinguish.

- **Case 1.** Suppose first that $x \in \mathbb{G}_L$. Clearly, in this case $Addr$ is a uniformly distributed group element in $\mathbb{G}_L$ regardless of the value of b. Thus,

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], x) = 1 \mid x \xleftarrow{R} \mathbb{G}_L] = \frac{1}{2}$$

- **Case 2.** Suppose now that $x \in \mathbb{G}_U$. This implies that $Addr$ is a valid stealth address in the view of $\mathcal{A}$. So the probability of algorithm $\mathcal{B}$ outputting 1 is exactly the same as $\mathcal{A}$ succeeds in $\mathsf{Game}_{\mathcal{A}}^{Anon}$, therefore

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], x) = 1 \mid x \xleftarrow{R} \mathbb{G}_U] = \frac{1}{2} + \epsilon_{Anon}$$

To summarize, the distinguishing advantage of $\mathcal{B}$ is calculated as

$$\Big|\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], x) = 1 \mid x \xleftarrow{R} \mathbb{G}_L] - \Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], x) = 1 \mid x \xleftarrow{R} \mathbb{G}_U]\Big| = \epsilon_{Anon},$$

which must be negligible according to $SM_{(\mathbb{G}_U,\mathbb{G}_L)}$ assumption. ■

5. CONCRETE CONSTRUCTIONS

5.1. Scheme based on 2^k-QR Problem

We now give a description of a concrete scheme based on 2^k-QR Problem. As described before, this FSA scheme FSA1 consists of algorithms (FSA1.Setup, FSA1.KeyGen, FSA1.AddrGen, FSA1. Test, FSA1.BTGen, FSA1.Retrieve).

- $\mathsf{Setup}(1^\lambda)$: Take as input a security parameter λ, and chooses an integer $k \geq 1$. Output public parameters $pp := k$.
- $\mathsf{KeyGen}(1^\lambda)$: Chooses random primes p_i, q_i satisfying $p_i = 2^k p_i' + 1$, $q_i = 2^k q_i' + 1$, where p_i', q_i' are also primes, and compute $N_i = p_i q_i$. Choose a quadratic non-residue $h_i \xleftarrow{R} \mathbb{J}(N_i) \backslash \mathbb{QR}(N_i)$. Set $pk_i := (N_i, h_i), sk_i = p_i$. Output key pair (pk_i, sk_i).

- AddrGen(pk_i, S): To generate a transaction address for user i, first choose a random $x \xleftarrow{R} \mathbb{Z}^*_{N_i}$. Using CRT, compute a joint integer such that

$$\begin{cases} Y \equiv h_i \cdot x^{2^k} \mod N_i \\ Y \equiv x^{2^k} \mod N_j, \quad \forall j,\ j \neq i \end{cases}$$

  Then it sets the one-time address as $Addr_{pk_i} = Y$; finally, the transaction address is (N, Y).
- Test($Addr, pk_i, sk_i$): Takes as input a transaction address $Addr = Y$, first compute $Y_i = Y \mod N_i$. Then compute $b = \left(\frac{Y_i}{p_i}\right)_{2^k} = Y_i^{\frac{p_i-1}{2^k}} \mod p_i$. If $b = 1$, user i rejects the transaction tied to $Addr$; otherwise, accepts and stores the asset into its wallet.
- BTGen($\{Addr_i\}_{i=1}^T$): Same as the generic construction, following the procedure shown in Section 4.
- Retrieve(BT, pk_i, sk_i): Same as the generic construction, following the procedure of Algorithm 1.

Computing the exact amount of incoming transactions: In this scheme, user i could calculate the total number of transactions intended for him from the root node $R = Y_{(1,2,\cdots,n)}$ very efficiently following the idea of [27]. To see this, first suppose the exact number is $n_i < 2^k$. Then according to CRT, we must have $R \equiv R_i = h_i^{n_i} \cdot (x_1 \cdot x_2 \cdots x_n)^{2^k} \mod N_i$, where $x_1, x_2 \cdots, x_n$ are random integers chosen by n different senders. If we view n_i as a k-bit integer with $n_i = \sum_{j=0}^{k-1} n_i^{(j)} \cdot 2^j$, where $n_i^{(j)} \in \{0, 1\}$, then the following must hold:

$$\left(\frac{R_i}{p_i}\right)_{2^\kappa} = \left(\frac{h_i^{n_i} X^{2^k}}{p_i}\right)_{2^\kappa} = \left(\frac{h_i^{\sum_{j=0}^{\kappa-1} n_i^{(j)} \cdot 2^j}}{p_i}\right)_{2^\kappa} = \left(\frac{h_i}{p_i}\right)_{2^\kappa}^{\sum_{j=0}^{\kappa-1} n_i^{(j)} \cdot 2^j} \mod p_i$$

for $1 \leq \kappa \leq k$. This allows for a bit-by-bit extraction of n_i using trapdoor p_i. For example, if we take $\kappa = 1$, then the least significant bit can be extracted as $n_i^{(0)} = 1$ if $\left(\frac{R_i}{p_i}\right)_2 = -1 \mod p_i$ or $n_i^{(0)} = 0$ if $\left(\frac{R_i}{p_i}\right)_2 = 1 \mod p_i$ since $h_i \in \mathbb{J}(N_i) \backslash \mathbb{QR}(N_i)$ and $\left(\frac{R_i}{p_i}\right)_2 = \left(\frac{h_i}{p_i}\right)_2^{n_i^{(0)}}$

5.2. Scheme based on p'-subgroup Decision Problem

We now give description of a concrete scheme based on p'-subgroup Decision Problem. This FSA scheme FSA2 consists of algorithms (FSA2.Setup, FSA2.KeyGen, FSA2.AddrGen, FSA2. Test, FSA2.BTGen, FSA2.Retrieve):

- Setup(1^λ) $\rightarrow pp$: Takes as input a security parameter λ, and outputs public parameters pp.
- KeyGen(1^λ) $\rightarrow$ (pk_i, sk_i): Chooses random safe primes $p_i = 2p_i' + 1, q_i = 2q_i' + 1$, and compute $N_i = p_i q_i$. Suppose that $N_i' = p_i' q_i'$. Randomly choose two elements $x, y \in \mathbb{Z}^*_N$ and set $g_i = x^2 \mod N_i, \hat{g}_i = y^2 \mod N_i$. Note that with high probability $ord(g_i) = ord(\hat{g}_i) = N_i'$. Compute $h_i = \hat{g}_i^{q_i'} \mod N_i$. Output a key pair $pk_i := (N_i, g_i, h_i), sk_i := p_i'$.
- AddrGen(pk_i, S): To generate a transaction address for user i, first it chooses a random $r_i \in \{0,1\}^\lambda$; second, it computes an integer $Y_i = g_i \cdot h_i^{r_i} \mod N_i$; third, it sets $Y_j = h_j^{r_j} \mod N_j$; according to CRT, compute a joint number Y such that

$$\begin{cases} Y \equiv g_i h_i^{r_i} \mod N_i \\ Y \equiv h_j^{r_j} \mod N_j, \quad \text{for } j \neq i \end{cases};$$

  fourth, it computes the user's one-time address $Addr_{pk_i} = Y$; finally, the transaction address is (N, Y).
- Test($Addr, pk_i, sk_i$): Takes as input a transaction address $Addr = Y$, first compute $Y_i = Y \mod N_i$. Then compute $b = Y_i^{p_i'} \mod N_i$. If $b = 1$, user i discards the transaction tied to $Addr$; otherwise, store the asset into its wallet.
- BTGen($\{Addr_i\}_{i=1}^T$): Same as the generic construction, following the procedure shown in Section 4.
- Retrieve(BT, pk_i, sk_i): Same as the generic construction, following the procedure of Algorithm 1.

5.3. Security Analysis

The following theorems illustrate the security of the proposed (fast) stealth address schemes.

Theorem 5.1. If the 2^k-QR assumption holds in group $\mathbb{Z}^*_{N_i}$, then the stealth address scheme FSA1 satisfies anonymity.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the anonymity of FSA1 with advantage ϵ_{Anon}. Then we can construct a PPT simulator $\mathcal{B}$ that solves the 2^k-QR problem with almost the same advantage.

Let $N = pq$ be a modulus such that $p = 2^k p' + 1, q = 2^k q' + 1$, $\mathbb{G}_U = G_{p'q'} \cdot G_{2^k}$ and $\mathbb{G}_L = G_{p'q'}$ as defined in Section 2.7. The inputs of $\mathcal{B}$ consist of $(N, [\mathbb{G}_U], [\mathbb{G}_L], h)$, the target of $\mathcal{B}$ is to distinguish that $h \xleftarrow{R} \mathbb{G}_U$ or $h \xleftarrow{R} \mathbb{G}_L$. $\mathcal{B}$ simulates the $\mathsf{Game}^{Anon}_{\mathcal{A}}$ as follows.

- **Setup**: $\mathcal{B}$ runs $pp \leftarrow$ Setup(1^λ) and ($pk_i := (N_i, h_i), sk_i := p_i$) $\leftarrow$ KeyGen($1^\lambda; r_i$) for all $i \in [n]$, where $n = \text{poly}(\lambda)$ and r_i is the randomness used in KeyGen. $\mathcal{C}$ sets $S = \{pk_1, \cdots, pk_n\}$ and initializes an empty set L_c. Finally, $\mathcal{B}$ sends (pp, S) to $\mathcal{A}$.
- **Query Phase**: $\mathcal{A}$ can adaptively query the following oracle:
  - Corrupting oracle **Corrupt**($\cdot$): On input an index $i \in [n]$ such that $pk_i \in S$, this oracle returns $sk_i = p_i$ and adds $pk_i = (N_i, h_i)$ to L_c.
- **Challenge**: $\mathcal{A}$ selects a public key $pk_{i^*} \in S \wedge pk_{i^*} \notin L_c$, where pk_{i^*} is implicitly set by $\mathcal{B}$ as $(N_{i^*} := N, h_{i^*} := h)$ and sends it to $\mathcal{B}$. $\mathcal{B}$ picks a random $x \in \mathbb{Z}^*_N$ and a random bit $b \in \{0, 1\}$. With the help of CRT, $\mathcal{B}$ computes an integer Y such that

$$\begin{cases} Y \equiv h_{i^*}^b \cdot x^{2^k} \mod N_{i^*} \\ Y \equiv x^{2^k} \mod N_j, \quad \forall j,\ j \neq i^* \end{cases}$$

  Then $\mathcal{B}$ sets $Addr^* = Y$ and then returns it to $\mathcal{A}$.
- **Guess**: Finally, $\mathcal{A}$ outputs a bit b' as its guess of b. If $b' = b$, $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0.

There are two cases to distinguish.

- **Case 1.** Suppose first that $h \in \mathbb{G}_L$. Clearly, in this case $Addr^* \mod N_{i^*}$ is uniformly distributed in $\mathbb{G}_L$ regardless of b. Thus,

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} \mathbb{G}_L] \leq \frac{1}{2}$$

- **Case 2.** Suppose now that $h \in \mathbb{G}_U$. This implies that $Addr^*$ $\bmod N_{i^*}$ is a well-formed stealth address implicitly targeted at user i* when $b = 1$ in the view of $\mathcal{A}$. If $\mathcal{A}$ can tell whether or not $Addr^*$ is intended for user i^*, so can the simulator distinguish between the case that $h \in \mathbb{G}_U$ and $h \in \mathbb{G}_L$. Therefore, the probability of algorithm $\mathcal{B}$ outputting 1 is exactly the same as $\mathcal{A}$ succeeds in $\mathsf{Game}_{\mathcal{A}}^{Anon}$, therefore

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} \mathbb{G}_U] = \frac{1}{2} + \epsilon_{Anon}$$

To summarize, the distinguishing advantage of $\mathcal{B}$ is calculated as

$$\left| \Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} \mathbb{G}_L] - \Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} \mathbb{G}_U] \right| \leq \epsilon_{Anon},$$

which must be negligible according to 2^k-QR assumption. ■

Theorem 5.2. If the p'-subgroup Decision assumption holds in group $\mathbb{Z}_{N_i}$, then the stealth address scheme FSA2 satisfies anonymity.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the anonymity of FSA2 with advantage ϵ_{Anon}. Then we can construct a PPT simulator $\mathcal{B}$ that solves the p'-subgroup decision problem with almost the same advantage.

Let N be a strong RSA modulus, $\mathbb{G}_U = G_{p'q'}$ and $\mathbb{G}_L = G_{p'}$ as defined in Section 2.7. The inputs of $\mathcal{B}$ consist of $(N, [\mathbb{G}_U], [\mathbb{G}_L], h)$, the target of $\mathcal{B}$ is to distinguish that $h \xleftarrow{R} \mathbb{G}_U$ or $h \xleftarrow{R} \mathbb{G}_L$. $\mathcal{B}$ simulates the $\mathsf{Game}_{\mathcal{A}}^{Anon}$ as follows.

- **Setup**: $\mathcal{B}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(pk_i := (N_i, g_i, h_i), sk_i := p_i) \leftarrow \mathsf{KeyGen}(1^\lambda; r_i)$ for all $i \in [n]$, where $n = \mathrm{poly}(\lambda)$ and r_i is the randomness used in KeyGen. $\mathcal{C}$ sets $S = \{pk_1, \cdots, pk_n\}$ and initializes an empty set L_c. Finally, $\mathcal{B}$ sends (pp, S) to $\mathcal{A}$.
- **Query Phase**: $\mathcal{A}$ can adaptively query the following oracle:
  - Corrupting oracle $\mathsf{Corrupt}(\cdot)$: On input an index $i \in [n]$ such that $pk_i \in S$, this oracle returns $sk_i = p_i$ and adds $pk_i = (N_i, g_i, h_i)$ to L_c.
- **Challenge**: $\mathcal{A}$ selects a public key $pk_{i^*} \in S \wedge pk_{i^*} \notin L_c$ and sends it to $\mathcal{B}$. pk_{i^*} is implicitly set by $\mathcal{B}$ as $(N_{i^*} := N, g_{i^*} =, h_{i^*} := h)$. $\mathcal{B}$ picks a random $r \in \{0,1\}^\lambda$ and a random bit $b \in \{0,1\}$. With the help of CRT, $\mathcal{B}$ computes an integer Y such that

$$\begin{cases} Y \equiv g_{i^*}^b \cdot h_{i^*}^r \mod N_{i^*} \\ Y \equiv h_j^r \mod N_j, \qquad \forall j,\ j \neq i^* \end{cases}$$

  Then $\mathcal{B}$ sets $Addr^* = Y$ and then returns it to $\mathcal{A}$.
- **Guess**: Finally, $\mathcal{A}$ outputs a bit b' as its guess of b. If $b' = b$, $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0.

There are two cases to distinguish.

- **Case 1.** Suppose first that $h_{i^*} \in G_{p'q'}$. Clearly, in this case $Addr^*$ $\bmod N_{i^*} = g_{i^*}^b \cdot h_{i^*}$ is uniformly distributed in $G_{p'q'}$ regardless of b.Therefore, in the view of $\mathcal{A}$ the value of b is perfectly hidden. The probability of algorithm $\mathcal{B}$ outputting 1 is exactly the same as $\mathcal{A}$ succeeds in $\mathsf{Game}_{\mathcal{A}}^{Anon}$; therefore,

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} G_{p'q'}] \leq \frac{1}{2}$$

- **Case 2.** Suppose now that $h_{i^*} \in G_{p'}$. This implies that $Addr$ is a well-formed address intended for user i* when $b = 1$ in the view of $\mathcal{A}$. To elaborate, this case is divided into two subcases considering the value of b:
  - **subcase 1:** $b = 0$. $Addr^*$ $\bmod N_{i^*} = h_{i^*}^r$ is a normal stealth address that does not take i* as the target recipient.
  - **subcase 2:** $b = 1$. $Addr^*$ $\bmod N_{i^*} = g_{i^*} \cdot h_{i^*}^r$ is a normal stealth address that takes i* as the target recipient.

From this observation, it can be deduced that the probability of algorithm $\mathcal{B}$ outputting 1 is exactly the same as $\mathcal{A}$ succeeds in $\mathsf{Game}_{\mathcal{A}}^{Anon}$; therefore,

$$\Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} G_{p'}] = \frac{1}{2} + \epsilon_{Anon}$$

To summarize, the distinguishing advantage of $\mathcal{B}$ is calculated as

$$\left| \Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} G_{p'q'}] - \Pr[\mathcal{B}(N, [\mathbb{G}_U], [\mathbb{G}_L], h) = 1 \mid h \xleftarrow{R} G_{p'}] \right| \leq \epsilon_{Anon},$$

which must be negligible according to p'-subgroup decision assumption. ■

6. IMPLEMENTATION AND EVALUATION

In this section, we implement and evaluate our concrete fast stealth schemes FSA1, FSA2 based on open-source library. The performance of these two schemes are almost the same and below we only show the results of 2^k-QR-based scheme. For simplicity, we denote this scheme by FSA.

6.1. Implementation

Our implementation works on a 2019 MacBook Pro (Intel Core i7 at 2.6 GHz, 16GB RAM at 2.667 GHz) running MacOS 11.5.1, using C++ language. The implementations are single-threaded. We use OpenSSL 1.1.1l open-source library for the generation of prime numbers and for big number operations over $\mathbb{Z}_N$.We implemented our schemes with parameter $|N| = 2048, 3072$, which is the bit-length of a single recipient's modulus N. Such parameter choices will provide 112-bit and 128-bit level of security.

6.2. Evaluation Results

We consider different configurations by varying the number of recipients, for $200 \leq n \leq 2000$, and we measure the time performance of the testing and searching operations for each configuration when transaction quantity consecutively grows. The results of our evaluation are shown in Figs 3 and 4. Every plot illustrates the performance trend of a given operation as the number of transactions T increases. More specifically, Fig. 3 shows the time of testing whether a transaction exists among a bunch of transactions, while Fig. 4 depicts the time required to locate the specific transactions.

We discuss the implementation results of our scheme and give time complexity comparison with mainstream ECC-based stealth address scheme applied in Monero (Monero SA for simplicity), the most popular privacy-centric cryptocurrency. We implement the

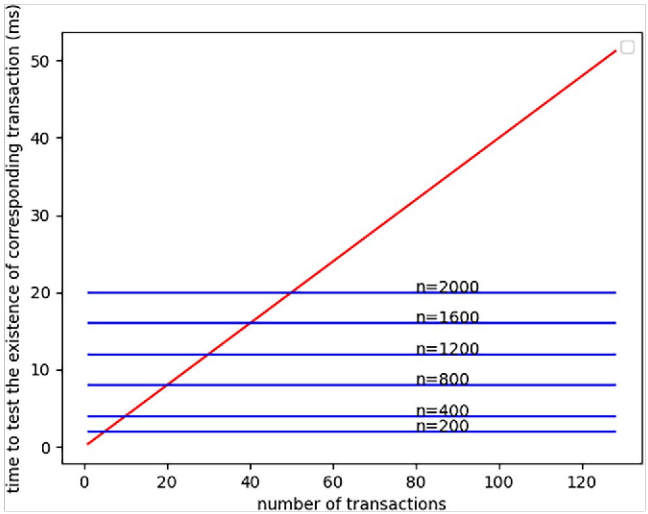

Figure 5. Comparing times to test the existence of transactions with varying number of recipients and transactions. The number of recipients n in the system is taken from the set {200, 400, 800, 1200, 1600, 2000}. The slash line represents the performance of Monero SA scheme, while the vertical lines stand for that of our FSA scheme in different settings.

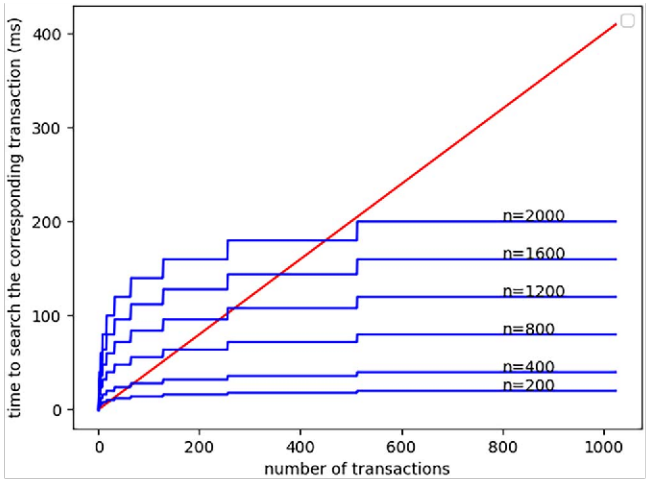

Figure 6. Comparing times to search corresponding transactions with varying number of recipients and transactions. The number of recipients n in the system is taken from the set {200, 400, 800, 1200, 1600, 2000}. The slash line reflects a linear search time over the entire transaction list of Monero SA scheme, while the curves reflect a logarithmic trend of FSA searching on the binary-tree built from BTGen algorithm.

Monero SA in the same environment, which is officially running on Ed25519 Curve with 128-bit level of security.

Recipient Computation. Our experiments show that in the Monero SA scheme the computation time of a recipient is independent of the number of recipients in the system and grows linearly with the number of transactions considering both the test of existence and the search process. In our FSA scheme the test process takes constant time, while the search time grows logarithmically with the number of transactions. As is also shown, in our scheme a recipient's computation time also scales moderately with respect to the number of recipients. Our FSA scheme obviously outperforms Monero SA scheme when taking the scale of transactions as the main impact factor. For example, it is about 70 ms in our scheme to search from $T = 600$ transactions for a system containing $n = 800$ recipients, while it is about 250 ms taken in Monero SA in the same recipient scale. It takes about 20 ms to search from $T = 1000$ transactions for $n = 200$ recipients, while it is about 400 ms taken in Monero SA scheme. When the value of n is fixed, the performance of our scheme tends to be stable. This is due to that the computation cost of a recipient is only related to the depth of underlying search tree $d = \log_2 T$, thus is almost the same when $T = 600$ or $T = 1000$. Our scheme is more suitable for a scenario where there are moderate volume of users but large scale of transactions.

6.2.1. *Sender Computation and Storage Overhead*

We assume that a sender is going to send a stealth address transaction to a recipient with anonymity set size 100, i.e. the sender chooses 100 users to compute an integer and send it on blockchain. We measure two parameter settings of our scheme which achieve different level of security (112,128 bit). The stealth address generation algorithm of our scheme triggers a process that requires running modulus-related exponentiation operations and a CRT combination process to derive the final stealth address. The computation overhead of the sender and the address size is thus proportional to the total number of recipients.

Table 1 gives concrete measurements of FSA in terms of key size, time to generate an address and stealth address size.

7. CONCLUSION AND DISCUSSION

In this paper, we have introduced the model of FSA that abstracts real-world recipient anonymous blockchain applications. We provide a formal definition in the well-formed indistinguishability-based framework, server-aided solutions that achieve this definition (in the semi-honest setting), and give implementation results. Our protocols achieve a logarithmic computational efficiency for the recipients, requiring only asymptotically minimal overhead. The workload of the server, however, is proportional to $O(n)$ per block for n recipients in the anonymity set, which limits the choice of the parameters of n. We leave it as a future work to explore techniques to improve the workload of the servers and the construction of a space-efficient FSA scheme.

The anonymity set is assumed to be the default full set containing all n registered recipients, which inevitably incurs an $O(n)$-sized value sent by a sender. It is possible to reduce sender's computation overhead by asking the sender to choose the set of anonymity only on a subset m out of n of all recipients. This gives us a similar trade-off between privacy and sender efficiency

Table 1. Sender computation cost and communication overhead of FSA

Scheme	Security Level	Key Size (bytes)	AddrGen (ms)	Address Size (KB)	Search Time of 1000 Tx (ms)
Monero SA	128	pk: 65 sk: 32	0.47	0.064	420.13
FSA-2048	112	pk: 256 sk: 128	0.39	24.9	17.41
FSA-3072	128	pk: 384 sk: 192	0.82	37.49	43.87

as in FMD. This modification could certainly save bandwidth in practice, but whether there are any vulnerabilities in real-world applications remains unclear and we believe it as an interesting direction of study.

DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author.

REFERENCES

1. Nakamoto, S. and Bitcoin, A. (2008) A peer-to-peer electronic cash system. *Bitcoin* https://bitcoin.org/bitcoin. pdf, **4**, 2.
2. Rivest, R.L., Shamir, A. and Tauman, Y. (2006) How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science*, pp. 164–186. Springer Berlin, Heidelberg.
3. Courtois, N. T. and Mercer, R. (2017) Stealth address and key management techniques in blockchain systems. *Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017*, Porto, Portugal, *February 19–21*, pp. 559–566. SciTePress.
4. Miers, I., Garman, C., Green, M., and Rubin, A. D. (2013) Zerocoin: Anonymous distributed e-cash from bitcoin. *2013 IEEE Symposium on Security and Privacy, SP 2013*, Berkeley, CA, USA, *May 19–22*, pp. 397–411. IEEE Computer Society.
5. Möser, M. *et al.* (2018) An empirical analysis of traceability in the monero blockchain. *Proc. Priv. Enhancing Technol.*, **2018**, 143–163.
6. Zheng, Y., Ye, H., Dai, P., Sun, T. and Gelfer, V. (2019) Confidential assets on mimblewimble. Cryptology ePrint archive. Report 2019/1435.
7. Beck, G., Len, J., Miers, I., and Green, M. (2021) Fuzzy message detection. *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 -19.*, pp. 1507–1528. ACM.
8. Madathil, V., Scafuro, A., Seres, I. A., Shlomovits, O., and Varlakov, D. (2022) Private signaling. *31st USENIX Security Symposium, USENIX Security 2022*, Boston, MA, USA, *August 10-12*, pp. 3309–3326. USENIX Association.
9. Brakerski, Z. and Goldwasser, S. (2010) Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference*, Santa Barbara, USA, *August 15-19*, pp. 1–20. Springer.
10. Yamamura, A. and Saito, T. (2001) Private information retrieval based on the subgroup membership problem. In *Information Security and Privacy*, pp. 206–220. Springer Berlin, Heidelberg.
11. Gjøsteen, K. (2005) Symmetric subgroup membership problems. In *Public Key Cryptography - PKC 2005*, pp. 104–119. Springer, Berlin Heidelberg.
12. Jager, T. and Schwenk, J. (2008) The generic hardness of subset membership problems under the factoring assumption. Cryptology ePrint archive. Report 2008/482.
13. Cao, Y., Wang, Y., Ding, Y., Guo, Z., Wu, Q. and Liang, H. (2023) Blockchain-empowered security and privacy protection technologies for smart grid. *Comput. Stand. Interfaces*, **85**, 103708.
14. Chien, I., Karthikeyan, P., and Hsiung, P. (2023) Peer to peer energy transaction market prediction in smart grids using blockchain and LSTM. *IEEE International Conference on Consumer Electronics, ICCE 2023*, Las Vegas, NV, USA, *January 6-8, 2023*, pp. 1–2. IEEE.
15. Annane, B., Alti, A. and Lakehal, A. (2022) Blockchain based context-aware CP-ABE schema for internet of medical things security. *Array*, **14**, 100150.
16. Kamruzzaman, M., Yan, B., Sarker, M.N.I., Alruwaili, O., Wu, M. and Alrashdi, I. (2022) Blockchain and fog computing in iot-driven healthcare services for smart cities. *J. Healthc. Eng.*, **2022**, 1–13.
17. Namasudra, S., Sharma, P., Crespo, R.G. and Shanmuganathan, V. (2023) Blockchain-based medical certificate generation and verification for iot-based healthcare systems. *IEEE Consumer Electron. Mag.*, **12**, 83–93.
18. Tsai, C. (2023) Supply chain financing scheme based on blockchain technology from a business application perspective. *Ann. Oper. Res.*, **320**, 441–472.
19. Oudani, M., Sebbar, A., Zkik, K., Harraki, I.E. and Belhadi, A. (2023) Green blockchain based iot for secured supply chain of hazardous materials. *Comput. Ind. Eng.*, **175**, 108814.
20. Gutoski, G. and Stebila, D. (2015) Hierarchical deterministic bitcoin wallets that tolerate key leakage. *Financial Cryptography and Data Security - 19th International Conference, ref20 2015*, San Juan, Puerto Rico, *January 26-30, Revised Selected Papers*, pp. 497–504. Springer.
21. Van Saberhagen, N. (2013). Cryptonote v 2.0.
22. Miller, A., Möser, M., Lee, K. and Narayanan, A. (2017) An empirical analysis of linkability in the monero blockchain. *CoRR*, abs/1704.04299.
23. Zheng, Z., Xie, S., Dai, H.-N., Chen, X. and Wang, H. (2018) Blockchain challenges and opportunities: a survey. *Int. J. Web Grid Serv.*, **14**, 352–375.
24. Yaga, D., Mell, P., Roby, N. and Scarfone, K. (2019) Blockchain technology overview. *CoRR*, abs/1906.11078.
25. Kurosawa, K. and Tsuji, S. (1990) A general method to construct public key residue cryptosystems. *Trans Ieice*, **73**, 1068–1072.
26. Zheng, Y. (2001) Identification, signature and signcryption using high order residues modulo an RSA composite. *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001*, Cheju Island, Korea, *February 13-15*, pp. 48–63. Springer.
27. Joye, M. and Libert, B. (2013) Efficient cryptosystems from 2^k-th power residue symbols. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Athens, Greece, *May 26-30*, pp. 76–92. Springer.
28. Xue, H., Li, B., Lu, X., Jia, D., and Liu, Y. (2013) Efficient lossy trapdoor functions based on subgroup membership assumptions. *Cryptology and Network Security - 12th International Conference*, Paraty, Brazil, *November 20-22.*, pp. 235–250. Springer.