# SoK: Privacy-Preserving Transactions in Blockchains

Foteini Baldimtsi[1,2], Kostas Kryptos Chalkias[1], Varun Madathil[3], and Arnab Roy[1]

[1]Mysten Labs, Palo Alto, CA
[2]George Mason University, Fairfax, VA
[3]Yale University, New Haven, CT

December 3, 2024

**Abstract**

Ensuring transaction privacy in blockchain systems is essential to safeguard user data and financial activity from exposure on public ledgers. This paper conducts a systematization of knowledge (SoK) on privacy-preserving techniques in cryptocurrencies with native privacy features. We define and compare privacy notions such as confidentiality, k-anonymity, full anonymity, and sender-receiver unlinkability, and categorize the cryptographic techniques employed to achieve these guarantees. Our analysis highlights the trade-offs between privacy guarantees, scalability, and regulatory compliance. Finally, we evaluate the usability of the most popular private cryptocurrencies providing insights into their practical deployment and user interaction. Through this analysis, we identify key gaps and challenges in current privacy solutions, highlighting areas where further research and development are needed to enhance privacy while maintaining scalability and security.

## 1 Introduction

Blockchain technology has transformed the digital landscape, enabling secure, decentralized systems for financial transactions, identity management, and beyond. However, the transparency inherent in most blockchain designs raises pressing concerns about transaction privacy. Unlike traditional financial systems, where transaction details are accessible only to trusted intermediaries, public blockchains expose all transactional data—sender and receiver identities (as pseudonyms) and transaction amounts—on an immutable ledger accessible to anyone. This level of exposure creates vulnerabilities for users, including the risk of surveillance, profiling, and exploitation by malicious actors (i.e. MEV (Maximal Extractable Value) attacks).

Achieving privacy for blockchain transactions is a highly complex task that must balance efficiency, scalability, and the level of privacy provided. Over the past decade, dozens of privacy-preserving mechanisms for blockchains have been proposed, all sharing the common goal of obscuring transaction details while maintaining the integrity and security of the blockchain. However, these solutions differ significantly in how they balance competing priorities, such as privacy guarantees, scalability, compatibility with existing systems, and additional features. Some mechanisms provide strong privacy guarantees but impose significant computational overhead or reduced throughput, making them less practical for high-volume networks. Others prioritize scalability through lighter-weight techniques, albeit with weaker privacy protections. Additionally, many solutions rely on advanced cryptographic assumptions, which, while powerful, can introduce risks and limitations of their own.

The result is a diverse and fragmented landscape of privacy-preserving techniques, each optimized for specific trade-offs and use cases. While some solutions prioritize strong anonymity, others emphasize efficiency or simplicity, reflecting the inherent difficulty of achieving all desired properties simultaneously. This diversity underscores the complexity of building privacy for blockchain transactions and highlights the need for a systematic understanding of the design space to navigate these trade-offs effectively.

## 1.1 Our Approach and Contributions

In this work, we aim to systematize the techniques and status of confidential transactions in blockchain systems. While the literature on privacy-preserving blockchains is extensive, it often lacks a unified framework to understand and compare the many definitions, techniques, and solutions proposed. Our work seeks to fill this gap by providing a structured analysis of the field, identifying key trends, challenges, and open problems and providing a blueprint for future designs based on the needs and properties of each system.

First, in Section 2.3 we systematize the various privacy definitions found in the literature. Privacy in blockchain transactions has been defined using a variety of techniques, such as indistinguishability game-based definitions and simulation based definitions in the Universal Composability (UC) model. These definitions vary in scope and rigor, leading to a fragmented understanding of privacy goals. We classify these definitions into three primary categories that capture the majority of existing systems' objectives, while also separately defining unlinkability, a property relevant to mixing techniques.

Next, in Section 3, we present the main cryptographic techniques used to design cryptocurrencies with confidential transactions. We separate between techniques used in UTxO-based blockchains from account-based blockchains since the underlying data model influences the design and implementation of the privacy mechanisms. We also briefly discuss the techniques used for mixing transactions as a standalone/external mixing tool, as well as the mechanisms to achieve regulatory compliance. As we also discuss in Section 1.2 below, our goal when discussing these mixing techniques is to understand their design, but our primary focus is on how such techniques are actually incorporated in cryptocurrencies as core functionalities and not external mechanisms to enhance their privacy.

In Sections 4 and 5 survey the state-of-the-art solutions, including both deployed systems and academic proposals. Our analysis is structured around UTxO- and account-based systems, comparing solutions based on key properties such as level of privacy achieved, transaction size, regulatory compliance, adoption level etc. Additionally, in Figure 2 we provide a decision-making graph as a blueprint to guide researchers and developers in selecting the most appropriate techniques or systems based on their specific requirements and constraints.

Beyond existing solutions, we identify open problems and challenges that remain unresolved, pointing to areas where further research is needed. Finally, in Section 6 we evaluate the usability of several popular systems, providing insights into the practical aspects of deploying and interacting with privacy-preserving blockchain technologies.

## 1.2 Scope

In this work, we focus exclusively on privacy-preserving techniques transactions on *permissionless* blockchain systems. We exclude permissioned systems, as they employ fundamentally different methodologies and trust models. Additionally, we do not cover frameworks designed for general private computation atop blockchains, such as Hawk [41] or Arbitrum [40], as our emphasis is on transaction-level privacy rather than broader computational privacy. While we briefly explain the functionality of the cryptographic building blocks used in these systems, such as zero-knowledge proofs, we do not get into a comparative analysis of these techniques or their underlying implementations. Furthermore, while we discuss the main techniques underlying external/standalone mixing mechanisms as Layer 2 (L2) protocols, we do not provide an extensive comparison of such services, as this could be a dedicated study in itself. Instead, we focus on mixing approaches that offer insights or techniques applicable to designing cryptocurrencies with native privacy-preserving features (L1s). The same holds for systems that support accountability and auditability.

### 1.3 Related Work

The topic of privacy in the blockchain setting has been extensively studied in the literature, underscoring its importance and highlighting the need for comprehensive SoKs. However, none of the existing works provides a systematic analysis of the current state of cryptocurrencies with native privacy, the techniques they employ, or a practical blueprint to help practitioners choose the best-suited schemes based on privacy levels and other characteristics. Furthermore, to the best of our knowledge, no prior SoKs have discussed the usability aspects of existing systems.

One of the most recent and relevant works is [1], which primarily focuses on systematizing cryptographic building blocks such as ZK proof systems and extends its scope to privacy on the blockchain beyond transactions, including topics like function privacy and private computation—areas outside the scope of our research. Similarly, [72] also systematizes ZKPs in the blockchain context. Two other works with overlapping scope are [34] and [2]. The former examines why traditional public ledgers lack anonymity but does not extensively cover or distinguish privacy techniques between UTxO and account-based blockchains. The latter focuses on comparing the privacy levels offered by various systems but neglects aspects like efficiency, scalability, and regulatory compliance. Additionally, works like [61] and [78] fail to distinguish between different privacy levels.

The scope of [26] aligns closely with ours, but it is now outdated, targets a less technical audience, and lacks the depth required for analyzing contemporary systems. Other related works focus on narrower domains, such as Bitcoin-style systems [27], smart contract privacy [43, 5], or privacy in DeFi applications [4]. Lastly, as noted in Section 1.2, we do not provide a comprehensive comparison of mixing services or techniques for accountability and auditability. Instead, we refer readers to existing SoKs in these areas: accountability [16] and mixing [3].

## 2 Background

We recall UTxO and Account-based blockchains, and then describe the notions of privacy that are applicable to them.

### 2.1 UTxO and Account-based Blockchains

We define a blockchain as a distributed ledger that securely and immutably records transactions. These transactions are validated through a consensus mechanism, ensuring the integrity of the ledger. Blockchains manage user accounts and transactions primarily through two models: the UTxO (Unspent Transaction Output) model and the Account-based model.

**UTxO Model.** The UTxO model, as implemented in Bitcoin [53], tracks discrete units of currency as outputs from previous transactions, which can be spent in future transactions. Each transaction consumes one or more UTxOs and generates new ones.

**Account-based Model.** The account-based model, used by Ethereum [10], tracks the balance of each account directly. This approach simplifies some transaction types but may introduce distinct privacy challenges due to the persistent tracking of account balances.

### 2.2 Cryptographic Primitives

We briefly review the most common cryptographic building blocks used to preserve privacy of transactions in the blockchain setting.

**Homomorphic Encryption.** A homomorphic encryption scheme allows computations to be performed directly on ciphertexts, producing an encrypted result that, when decrypted, matches the result of operations performed on the plaintext. More formally, given a homomorphic encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, let $C_1 = \mathsf{Enc}(pk, m_1)$ and $C_2 = \mathsf{Enc}(pk, m_2)$ be the encryptions of plaintexts $m_1, m_2$. Then, computing $\mathsf{Dec}(\mathsf{Eval}_f(C_1, C_2)) = f(m_1, m_2)$, where $f$ is a function applied to the plaintexts. Depending on the scheme, it may support either specific functions (such as addition or multiplication) or arbitrary computations (as in fully homomorphic encryption). Homomorphic encryption is used to conceal account balances or transactions on the blockchain, and a commonly used scheme is the ElGamal encryption scheme which can be either additively or multiplicatively homomorphic.

**Commitments.** A (non-interactive) commitment scheme consists of the following algorithms:

- $\mathsf{Gen}(1^\lambda)$ is an efficient randomized algorithm that outputs public parameters $\mathsf{pp}$.
- $\mathsf{Com}(\mathsf{pp}, m, r)$ is an efficient deterministic algorithm that takes as input the public parameters, a message $m$, and randomness $r$. It outputs a commitment to $m$ which we denote as $\mathsf{C}_m$.
- $\mathsf{Open}(\mathsf{pp}, m, r, \mathsf{C}_m)$ is a deterministic algorithm which on input an opening $(m, r)$ and a commitment $\mathsf{C}_m$, outputs a bit $b$ that indicates whether the commitment opening was successful.

A commitment scheme should be *hiding*, i.e. $\mathsf{C}_m$ should not reveal any information about $m$ and *binding*, i.e. it should be hard for the committer to find $m'$ such that $\mathsf{Com}(\mathsf{pp}, m, r) = \mathsf{Com}(\mathsf{pp}, m', r')$ with $m' \neq m$. There also exist commitment schemes with homomorphic properties. Pedersen commitments [60], are additively homomorphic, meaning that given two commitments $\mathsf{C}_m = \mathsf{Com}(\mathsf{pp}, m, r)$ and $\mathsf{C}_{m'} = \mathsf{Com}(\mathsf{pp}, m', r')$ to messages $m$ and $m'$, it is possible to compute a new commitment $\mathsf{C}_{m+m'} = \mathsf{C}_m \cdot \mathsf{C}_{m'}$ (or $\mathsf{C}_{m+m'} = \mathsf{Com}(\mathsf{pp}, m + m', r + r')$) without revealing $m$ or $m'$. When commitments are used to represent account balances and transaction amounts, additive homomorphism enables updating values in a privacy preserving manner.

**Zero Knowledge Proofs.** A zero-knowledge (ZK) proof $\pi$ enables a Prover $P$ who holds some private witness $w$ for a public instance $x$ and an NP-relation $R$, to convince a Verifier $V$ that some property of $w$ is true i.e. $R(x, w) = 1$, without $V$ learning anything more.

A zero-knowledge proof system between $P$ and $V$ for an NP relation $R$ must satisfy the following properties:

- *Completeness:* If $R(x, w) = 1$ and both players are honest $V$ always accepts.
- *Soundness:* For every malicious and computationally unbounded $P^*$, there is a negligible function $\epsilon(\cdot)$ s.t. if $x$ is a false statement (i.e. $R(x, w) = 0$ for all $w$), after $P^*$ interacts with $V$, $\Pr[V \text{ accepts}] \leq \epsilon(|x|)$.
- *Zero Knowledge:* For every malicious PPT $V^*$, there exists a PPT simulator $\mathcal{S}$ and negligible function $\epsilon(\cdot)$ s.t. for every distinguisher $D$ and $(x, w) \in R$ we have $|\Pr[D(\text{View}_{V^*}(x, w)) = 1] - \Pr[D(\mathcal{S}) = 1]| \leq \epsilon(|x|)$.

A ZK proof is called non-interactive (NIZK), if no interaction is required between the prover and the verifier during the creation of the proof. Then, the Prover given $(x, w)$ just sends one message $\pi$ to the Verifier and the Verifier outputs $0/1$ based on $(x, \pi)$.

NIZKs have been one of the most popular cryptographic tools to achieve transaction privacy in the blockchain, especially when they support *succinct* proofs (i.e. when the length of the proof is much smaller than the size of the statement being proved). However, succinct NIZKs typically come with the requirement of an elaborate *trusted setup* process that generates a common reference string (CRS) used by all the proofs and verifications. This CRS can be either created by a trusted third party, or via an MPC protocol. In either case, it is important that the random coins used to generate the CRS are never revealed. There also exist constructions of NIZKs with *transparent setup* which essentially replace the need for a CRS with the use of an idealized model such as the Random Oracle. Examples of NIZKs used in production include - Bulletproofs [9], or zero-knowledge succinct arguments of knowledge (zk-SNARKs) such as Halo2 [7].

**Accumulators and Merkle Trees.** Cryptographic accumulators are compact data structures that allow the succinct and binding representation of a set of elements by a single short value while supporting efficient membership proofs. More formally, an accumulator scheme $(\mathsf{Gen}, \mathsf{Add}, \mathsf{MemProofCreate}, \mathsf{Verify})$ enables the inclusion of an element $x$ into an accumulator $\mathsf{acc}$ by running the $\mathsf{Add}$ algorithm, while $\mathsf{MemProofCreate}$ creates a membership proof $\pi$. The verification algorithm $\mathsf{Verify}(\mathsf{acc}, x, \pi)$ is used to confirm membership without access to the accumulated set. The basic security property of accumulators is *soundness* which states that for every element *not* in the accumulator it is infeasible to prove membership. Finally, some accumulators can also additionally support deletion of elements, as well as non-membership proofs.

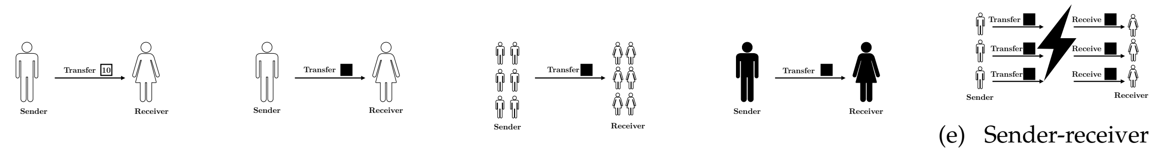
*Merkle trees* are a special type of cryptographic accumulators, structured as a binary tree where each leaf node contains a hash of data, and each internal node contains a hash of its child nodes. A Merkle root, derived from hashing all leaf nodes up to the root, succinctly represents the entire set of data, while a membership proof for a leaf node $x$, essentially consists of the path from the $x$ to the root of the tree.

Merkle trees and other types of cryptographic accumulators (based on RSA, Bilinear Pairings or Lattice assumptions) have been extensively used in the blockchain space [51, 15, 18, 59, 6] for scalability reasons in order to minimize storage and computational overheads and they can be composed with privacy tools, such as ZK proofs, in order to to balance scalability with privacy and confidential transactions.

## 2.3 Notions of Privacy

In this section we provide a systematization of different notions of privacy that can be achieved for blockchain transactions. Let $T = (\mathsf{S}, \mathsf{R}, \mathsf{x})$ denote a blockchain transaction where $\mathsf{S}$ stands for the Sender, $\mathsf{R}$ for the Receiver and $\mathsf{x}$ for the transaction amount. A transaction *without* any privacy support, reveals the Sender, Receiver and the transaction amount. We note that when we say that the Sender/Receiver of a blockchain transaction is revealed, it typically refers to the exposure of their address rather than their actual identity. This is often called *pseudoanonimity*. Given the extensive research on the limitations of pseudoanonimity [50], we do not consider this approach to offer any meaningful level of privacy.

We proceed with defining different privacy flavors starting with the weakest[1].



(a) No anonymity (b) Confidentiality (c) $k$-anonymity (d) Full anonymity (e) Sender-receiver Unlinkability

Figure 1: Representation of different privacy levels

- **Confidentiality:** Transaction confidentiality ensures that the transaction amount is hidden from all parties except the involved sender and receiver, i.e. no adversary can distinguish between two transactions with different amounts except with negligible probability. (This is often referred in the literature as a *confidential transaction*).

However, confidentiality alone is a relatively weak privacy notion, and stronger privacy concepts have been introduced to provide stronger guarantees, including hiding the identities of the sender and receiver in a transaction. However, there is no universally accepted definition of privacy that captures all the properties of the various schemes proposed in the literature. Different works use distinct techniques to define privacy, such as indistinguishability-based security notions

[1]We note that schemes that support stronger privacy flavors, should still guarantee that an adversary cannot "overspent". We consider this property, often called *balance* or *overdraft safety* a security property and do not explicitly discuss it here.

or simulation-based security notions (where the first are generally considered weaker than the second). A commonly referenced term for "strong" privacy is that of *ledger indistinguishability*, but we observe that it is interpreted differently across the literature, capturing varying properties depending on the features of the underlying protocol. In this section, we clarify these differences by distinguishing between k-anonymity and full anonymity as key privacy properties, since the techniques required to achieve each are fundamentally different.

- **k-Anonymity:** A blockchain achieves $k$-anonymity if, for any transaction $T = (\mathsf{S}, \mathsf{R}, \mathsf{x})$ , the sender or receiver cannot be distinguished from a *set* of at least $k$ other possible senders or receivers. A bit more formally, for a sender $S$ in a transaction, the anonymity set is defined as a group of $k$ users, such that for any adversary $\mathcal{A}$, the probability of correctly identifying the true sender from this set is at most $1/k$. The same definition applies to the receiver's anonymity in the transaction. If both the size of receiver and sender sets is $k$, then we simply say that the scheme achieves $k$-anonymity. Else, if the scheme only achieves sender or receiver side anonymity, we call it $k_S$-anonymity and $k_R$-anonymity respectively.
- **Full Anonymity:** A blockchain transaction provides full anonymity if sender, receiver and transaction amount, i.e. $(\mathsf{S}, \mathsf{R}, \mathsf{x})$ are all hidden. In a fully anonymous blockchain payment scheme, all transactions are completely obfuscated and any given transaction is indistinguishable from others on the blockchain. The adversary $\mathcal{A}$ cannot, with more than negligible probability, determine the identities of the sender or receiver or distinguish them from the full set of blockchain participants. Additionally, an adversary cannot even tell whether two transactions are the same or if the same sender is involved in multiple transactions.
  In some works, i.e. in ZCash [68], this property is described as an indistinguishability game where an adversary cannot distinguish between two versions of the ledger that differ in at least one transaction with non-negligible probability. Assuming that there are $n$ distinct addresses on the blockchain, one can consider full anonymity to be equivalent to $k$-anonymity for $k = n$.

The previous privacy notions typically refer to the level of privacy of an cryptocurrency with embedded privacy features. However, there also exist a series of "external" techniques, called *mixing mechanisms*, that operate on top of an cryptocurrency (without native privacy features) and naturally support a slightly different privacy flavor which is often referred to as *unlinkability*. Thus we also define the following property:

- **Sender-Receiver Unlinkability:** Sender-receiver unlinkability ensures that no adversary can link a *particular* sender to a specific receiver in a transaction. We define unlinkability in terms of an interaction multi-graph as in [33, 74]. Formally, given a set of transactions $T_i = (\mathsf{S_i}, \mathsf{R_i}, \mathsf{x_i})$, an adversary $\mathcal{A}$, observing the interaction multi-graph of senders and receivers, cannot, with more than negligible probability, determine whether any two transactions $T_i$ and $T_j$ involve the same sender-receiver pair. This definition is rooted in the concept of interaction multi-graphs, which map payments from senders to receivers, with edges labeled by the epochs in which the transactions occurred. Sender-receiver unlinkability requires that all compatible interaction graphs—those consistent with the observable transaction data—are equally likely. Thus, the anonymity set is defined by the number of such compatible interaction graphs, ensuring that even if the identities of the sender and receiver are individually known, their transactional relationship remains hidden.

*Relation to full and k-anonymity.* The notion of sender-receiver unlinkability is weaker than full anonymity but shares some similarities with $k$-anonymity, though the two are distinct. Sender-receiver unlinkability focuses on protecting the *relationship* between a sender and receiver, ensuring their interactions remain untraceable even if their individual identities are exposed. In contrast, $k$-anonymity safeguards the *identities* of senders or receivers by embedding them within a fixed anonymity set of size $k$. While $k$-anonymity relies on an explicitly defined and fixed anonymity set (e.g., $k = 10$), sender-receiver unlinkability derives its anonymity set implicitly, based on the number of compatible interaction graphs, which depends on the transaction network's complexity rather than a predefined value.

### 2.4 Transaction and User Level Regulation

Blockchain schemes enhanced with privacy features like k-anonymity and full anonymity, often raise concerns for regulatory and law-enforcement reasons. At first glance, such privacy-preserving systems may appear incompatible with regulatory requirements. To address these concerns, certain desired properties—namely, auditability and accountability as defined in [16]—are introduced to enable compliance without compromising privacy.

- **Auditability:** This property ensures that an external auditor, with access to the public ledger, can provably obtain specific information required for auditing (e.g., the participants in a transaction). This process can be interactive, requiring the consent of the audited parties, or non-interactive, allowing the auditor to access the information directly.
- **Accountability:** This property allows the system to enforce predefined policies. For instance, it could reject transactions that do not comply with a spending limit (general accountability) or automatically disclose private information to a designated authority when certain conditions are met (designated party accountability).

## 3 Common Techniques

In this section, we will present the general blueprint followed by the solutions for achieving privacy in the account-based and UTxO settings.

### 3.1 Stealth Addresses

Stealth addresses were first introduced by Peter Todd [69] to improve the privacy properties of transactions on the Bitcoin blockchain. This technique does not require any changes to the underlying protocol and can be implemented on top of existing non-privacy-preserving blockchains like Bitcoin or Ethereum. In these schemes, the recipient of a transaction generates a one-time address that is not linked to their public address. The basic stealth address protocol works as follows: Let the receiver have the following public and private key pair: $(pk_R, sk_R)$. Note that $pk_i = g^{sk_i}$. The two parties engage in an ephemeral key exchange scheme as follows: The sender generates fresh keys $(R = g^r, r)$ and computes the shared secret as $s = H(pk_R^r)$. Note that $s = H(g^{sk_R \cdot r}) = H(R^{sk_R})$. The sender then sends a transaction to $P = pk_R \cdot g^s$. Since the receiver can compute $s$ independently, given $R$, they can spend this transaction using the secret key $sk_R + s$. Because the receiver needs their secret key online to identify transactions, it is common to use a viewing key to locate transactions and a separate spending key to spend them. This approach is known as the Dual Key Stealth Address Protocol. It also allows the receiver to outsource transaction tracking to an external service securely. Stealth addresses are currently used in Ethereum [77], Bitcoin [69], the Aztec Network [75], Monero[62] and many other cryptocurrencies.

### 3.2 Privacy-preserving Account-based Cryptocurrencies

**The state.** As mentioned above the state of the blockchain can be abstracted as a list of account-balance pairs in account-based cryptocurrencies. In the privacy-preserving setting, the balance is hidden, using either a homomorphic commitment scheme or a homomorphic encryption scheme.

**Achieving Confidentiality.** Recall that a transaction in account-based cryptocurrencies has the form $(S, R, x)$. To achieve confidentiality, the value of the transaction $x$ is replaced by two ciphertexts $C_S = Enc(pk_S, -x)$ and $C_R = Enc(pk_R, +x)$ which debit the sender's account and credit the receiver's account with value $x$, respectively. The transaction also includes a zero-knowledge proof, typically proving that the sender's balance is greater than $x$ and $C_S$ and $C_R$ encrypt the same values with a difference in signs. The validator upon receiving this transaction will verify the zero-knowledge proof and then update the state by homomorphically adding $C_S$ to the sender's encrypted balance and adding $C_R$ to the receiver's encrypted balance. A user can also convert their private funds back to public funds via a *burn* transaction. The user proves that they know the secret key corresponding to their account in zero-knowledge.

**Achieving $k$-Anonymity.** To achieve $k$-anonymity, the typical idea is to include $k$ fresh dummy senders, receivers and ciphertexts in the transactions. These dummy senders and receivers are other accounts registered in the system. The ciphertexts corresponding to these accounts are simply encryptions of zero. The zero-knowledge proof now proves that only a pair of ciphertexts encrypt some value $x$ and $-x$ and all other ciphertexts are encryptions of zero. The validator as before verifies the zero-knowledge proof and then updates the state by homomorphically updating all the account balances that are indicated in the transaction. The correctness guarantee is that only the sender and receiver's balances are updated with some value, while the other account balances are updated with zero, but the ciphertext is updated. By the semantic security property of the underlying encryption scheme, the adversary cannot distinguish between a ciphertext that encrypts zero and a ciphertext encrypting the value of the transactions.

## 3.3 Privacy-preserving UTxO based Cryptocurrencies

**The State.** In the UTxO setting the state is typically the list of all unspent transactions. In the privacy-preserving setting, this corresponds to unspent coin commitments. When a coin is spent, a unique serial number corresponding to the coin is revealed. This ensures that the same coin is not spent twice. The list of all serial numbers also contribute to the state of the blockchain.

**Achieving confidentiality and anonymity.** As mentioned above, coins are represented by cryptographic commitments. These commitments hide the value of the transaction. More specifically, these systems have two kinds of transactions - Mint and Pour. A Mint transaction allows a user to create a private coin from a coin that has no privacy. Mint creates a commitment to the value of the public coin it is spending. Associated to this coin is also a random seed, that the user stores locally. The coin commitment becomes part of a list of coin commitments. A Pour transaction allows a user to transfer value from one private coin to another, which is accomplished by creating a new coin commitment and revealing a serial number which is computed by evaluating a pseudorandom function on the random seed. The user attaches a zero-knowledge proof (a NIZK) to prove that the serial number corresponds to one of the coin commitments in the list of coin commitments. It also proves that the serial number and the commitment was computed correctly using the user's keys which are in a set of all keys. Typically these proofs are succinct and require a trusted setup. The Pour transactions also include a public component that allows users to pay fees or alternatively to convert their private coins to public coins.

**Achieving a weaker notion of anonymity.** There is also another direction to achieve privacy-preserving UTxO that makes use of ring signatures. This avoids the use of a trusted setup but achieves $k$-anonymity, which is a weaker notion of anonymity.

## 3.4 Mixing Techniques

A series of mixing techniques have been proposed in the literature which can be broadly categorized into three main types based on their operational models: *peer-to-peer (P2P) mixing and centralized tumblers, and smart contract-based mixers*. Below, we explore these categories, focusing on the methods they employ to achieve sender-receive unlinkability.

**P2P Mixing.** P2P mixing involves users collaboratively combining their transactions into a single transaction, as in CoinJoin [28][2], without relying on intermediaries. Such schemes, typically use

[2]CoinJoin is a classic decentralized mixing protocol in UTxO-based blockchains that combines inputs and outputs from

shuffling techniques, multi-signatures and in some cases even stealth addresses (to get confidential transactions) to break the linkage between senders and receivers, making all possible input-output mappings equally likely. The challenge with such designs is the bootstrapping problem —the difficulty in identifying and coordinating a sufficiently large group of participants. Larger groups offer better anonymity but lead to poor scalability and increased protocol complexity.

**Centralized Tumblers.** In these approach, a tumbling service, which acts as an intermediary, receives and redistributes funds in order to obscure the transaction links. Schemes that build untrusted tumblers, employ techniques such as blind signing, cryptographic puzzles and ZK proofs in order to achieve security and privacy against the tumbler. To increase scalability, some schemes additionally use payment-channels to remove some of the interactions between senders/receivers and the tumbler off-chain. The challenge in such designs is the increased dependence in the availability of centralized parties which might also be vulnerable to regulatory scrutiny or shutdowns.

**Smart contract-based mixers.** Such designs take a decentralized approach, automating the mixing process through smart contracts. A common approach involves users depositing funds into a smart contract, which records cryptographic commitments representing the deposited amounts. These commitments are later used to generate unlinkable withdrawals, where the link between the deposited funds and the withdrawn amounts is cryptographically broken. Zero-knowledge proofs or trusted execution environments (TEEs) can also be to achieve unlinkability by allowing users to prove valid withdrawal claims without exposing their original deposit information. Accumulators and Merkle-trees can be used for increased scalability. A challenge with such approaches lies in the careful design of the underlying smart contracts since vulnerabilities in the contract code can compromise user privacy or security of funds.

In terms of compatibility with UTxO and account based blockchains, P2P mixing is generally more compatible with UTxO-based blockchains while the other two approaches can be applied to both UTxO and account-based settings. Still, across all the mixing methods, the scalability of the mixing process and the potential regulatory scrutiny remain significant challenges, requiring careful design to balance privacy, efficiency, and trust.

## 3.5 Mechanisms for Auditability and Accountability

To enable auditability and accountability in privacy-preserving blockchain systems, cryptographic techniques have been developed that balance regulatory compliance with user privacy. These mechanisms allow transactions to be verified without exposing sensitive information about the parties involved. We identify two primary techniques to achieve auditability while preserving privacy and refer the reader to [16] for a more detailed discussion of these methods.

**Viewing Keys** support *auditability* by allowing trusted entities, typically referred to as *auditors*, to selectively access specific transaction details—such as amounts or the identities of the sender and receiver—without granting them full access to the entire blockchain. This selective transparency ensures that sensitive information can be shared with auditors for verification purposes while maintaining the privacy of other users and unrelated transactions.

**ZK-proofs** enable *accountability* by allowing users to prove that a transaction satisfies specific policies—such as "the transaction amount is less than $10,000"—without revealing any additional transaction details. These proofs ensure that predefined rules are enforced and that compliance with regulatory requirements is provable, while still protecting the privacy of the involved parties and the transaction content.

multiple users into a single transaction and thus achieves sender-receiver unlinkability.

# 4 Privacy-preserving transactions in the UTxO model

In this section we will present privacy-preserving schemes in the UTxO setting.

## 4.1 Confidentiality and Full Anonymity

We will first look at works that achieve confidentiality and full anonymity.

**Zerocoin and Zerocash.** Miers et al present Zerocoin [52], the first fully anonymous UTxO proposal in the literature. Following the blueprint outlined in Section 3.3, Zerocoin has two transactions called *mint* and *spend*, where the *mint* transaction allows a user to convert a bitcoin into a private coin. A private coin is simply a commitment to a serial number. To spend this coin, the user reveals the serial number and proves that this is a valid opening to one of the commitments in the list of minted coins. They use Pedersen commitments to commit to the serial numbers and the corresponding zero-knowledge proofs are constructed by first accumulating the set of commitments of all minted zerocoins, and then proving knowledge of the corresponding commitment randomness and membership in this set. The Zerocash protocol [68] extends this work by allowing for coins to have multiple denominations. The coin commitment is computed as follows : 1) commit to the address of the user $a_{pk}$ and a random seed $\rho$. Looking ahead, this $\rho$ will be used to compute the serial number that ensures that the coin is spent only once. 2) the above-computed commitment along with the value of the transaction are committed to compute the final coin commitment. These commitments are then collected to form a commitment tree. In Zerocash a tree can have only $2^{32}$ leaves, which aids them in proof generation. Both Zerocash and Zerocoin make use of a trusted setup to compute the zero-knowledge proofs that aid them in computing short proofs. Zerocash has been implemented as Zcash [19] and is a popular cryptocurrency today. The specifications for Zcash are continuously updated and can be found at [80]. A version of Zerocash with accountability was proposed by Garman et al [25].

**Curve Trees and $\mathbb{V}$Cash.** Campanelli et al [13] present Curve Trees which improve upon the state of the art for practical zero-knowledge for set membership. Notably, this finds applications in anonymous payments where users need to prove that they know a coin in the set of unspent coin commitments. In this work, they propose the use of a new accumulator scheme (dubbed Curve Trees) that does not require any trusted setup. Using these Curve Trees, they design a fully anonymous and efficient UTxO payment scheme called $\mathbb{V}$Cash. Their construction allows for batch verification but has slightly larger transaction size and longer proving and verification times.

## 4.2 Confidentiality and $k$-anonymity

**CryptoNote and later versions.** CryptoNote [76] is a cryptographic protocol that serves as the foundation for several privacy-focused cryptocurrencies, including Monero. Introduced in 2012 by Nicolas van Saberhagen, CryptoNote emphasizes transaction privacy and untraceability. Using ring signatures, the sender's identity is hidden by mixing their transaction with others, and one-time keys, which ensure that each transaction is linked to a unique public key. RingCT [56] proposed by Noether et al, presented a new type of ring signature termed Multi-layered Linkable Spontaneous Anonymous Group signature which allows for hidden amounts as well on top of sender and receiver addresses. RingCT2.0 [71] proposed by Sun et al, improves on RingCT by making the transaction size independent of the number of groups of input accounts included in the generalized ring. The original RingCT suffers a linear growth with the number of groups. But RingCT2.0 has the downside that it relies on a trusted setup.

Monero [62] a deployed blockchain, launched in 2014, adopted and expanded on the CryptoNote protocol and then the RingCT protocol to enhance user privacy. Triptych [55] is a ring signature construction proposed by the Monero Research Lab in January 2020. It offers logarithmic-sized linkable ring signatures, allowing for larger anonymity sets without significantly increasing transaction size. This allows users to include more decoy outputs in their transactions, making it more challenging to trace the true sender. Triptych achieves this by enabling efficient verification times, even with expanded ring sizes, thereby enhancing Monero's privacy features.

**Omniring.** Lai et al [42] present the first formal security definition for RingCT as a cryptographic primitive. They propose a protocol for RingCT termed Omniring, which does not require a trusted setup or pairing-friendly elliptic curves, has a proof size logarithmic in the size of the ring, and allows sharing the same ring between all source accounts in a transaction, thereby enabling significantly improved privacy level without sacrificing performance.

**Lelantus and Lelantus Spark.** Jivanyan [38, 39] presented the Lelantus protocols that are currently implemented as Firo [22] which is a popular privacy-preserving cryptocurrency. They modify the idea of Zerocoin to allow denominations of arbitrary value. More specifically the coin commitment is a commitment to a serial number and value. The receiver of the transaction provides a shielded address to the sender, which hides the identity of the receiver. To hide the sender's identity they use one-out-of-many proofs to prove that the serial numbers computed belong to one of the commitments in the commitment list. They do not make use of any trusted setup and use Bulletproofs for the zero-knowledge proofs and achieve logarithmic-sized proofs (in the anonymity set). Lelantus Spark [39] improves on Lelantus by introducing one-time addresses for the recipient, thereby not requiring the recipient to announce public addresses.

Finally, we remark that the Firo blockchain will be extending [63] the Lelantus Spark protocol to use curve trees [13] instead of one-out-of-many proofs to achieve full anonymity.

**Open Problem**: Can we construct an efficient anonymous UTxO-based scheme such that the state size is sub-linear (or constant) in the total number of transactions?

## 4.3 Confidentiality and Sender-receiver Unlinkability

In this section, we will present techniques that achieve sender-receiver unlinkability in the UTxO model. We start by describing representative standalone mixing services for the first 2 categories as defined in Section 3.4 and then we describe Mimblewimble and Dash which are cryptocurrencies with native privacy support that incorporate mixing techniques to achieve privacy.

**P2P Mixing.** CoinShuffle [66] and its improved version CoinShuffle++ [67] are characteristic examples of P2P mixing protocols. In these protocols, peers $p_1, p_2, \ldots, p_n$ collaboratively shuffle transaction outputs using pairwise shared keys $k_{ij}$ between each pair of participants $p_i$ and $p_j$. These shared keys, established through a Non-Interactive Key Exchange (NIKE) protocol, are used to securely encrypt outputs, ensuring that the link between inputs and outputs is obfuscated. After the shuffling phase, all participants sign the final transaction individually. To prevent malicious behavior, such as refusing to sign, CoinShuffle++ integrates a robust blame mechanism that allows honest participants to identify and prove the misbehavior of any disruptive participant without revealing sensitive information. One more variation of the scheme, ValueShuffle [65] additionally protects the transaction amounts.

**Centralized Tumblers.** Tumblebit [33] is a representative example of a centralized mixing service. It uses an intermediary, the *Tumbler*, to facilitate anonymous payments between a sender $S$ and a receiver $R$. The protocol operates in three phases: *Escrow*, *Puzzle-Solving*, and *Payment*. In the *Escrow Phase*, $S$ locks a payment of value $v$ in an address controlled jointly with the Tumbler, while $R$ creates a deposit address. During the *Puzzle-Solving Phase*, the Tumbler sends cryptographic puzzles $\mathsf{Puzzle}_i$ to $S$, who solves them and submits solutions $\mathsf{Sol}_i$, secured via hash commitments $\mathsf{H}(r)$. Finally, in the *Payment Phase*, $R$ receives the payment without being linked to $S$ (and achieving sender-receiver unlinkability). Tumblebit is compatible with Bitcoin and was implemented in a proof-of-concept wallet called "Breeze Wallet" by the Stratis blockchain platform. [73] presented $A^2L$, which improves Tumblebit by using adaptor signatures and randomizable puzzles and improves communication complexity.

**Mimblewimble.** Mimblewimble is currently deployed as part of Litecoin and MimblewimbleCoin. The first proposal was by an anonymous author and Poelstra [36] and was later formally analyzed by Fuchsbauer et al [23]. The Mimblewimble protocol combines three ideas - using Pedersen commitments for confidential transactions, using CoinJoin and transaction cut-through [29] - which is the idea that if a transaction spends an output $tx0_1$ and creates $txo_2$, which is then spent by another transaction creating $txo_3$, then this should be equivalent to a "cut-through" transaction spending $txo_1$ and directly creating $txo_3$. This actually allows users of the blockchain to not keep track of all unspent transactions hence resolving the problem of an ever-growing state that is common with all UTxO based systems. On the downside, the sender and the receiver of a transaction were required to interact with each other for the transaction to be spent by the receiver. Fuchsbauer et al [24] recently proposed non-interactive Mimblewimble transactions that allows transactions to be spent without any interaction.

**Dash.** *Dash* [30] is a UTxO-based cryptocurrency with native privacy features that essentially deploys CoinJoin. Dash has a set of designated nodes, called the *masternodes* which coordinate the set of transactions that will go through a CoinJoin. Given that there is no privacy against a masternode, Dash allows a user to specify the number of mixing iterations, through multiple masternodes, in order to increase the anonymity set size. Of course, this reliance on masternodes introduces a degree of trust, as these nodes must not retain logs of mixing activities and collude with each other. Thus, we say that Dash only achieves sender-receiver unlinkability at best (assuming no masternode collusion). We note that, unlike fully private cryptocurrencies, Dash's privacy features are optional, allowing users to choose whether to enable PrivateSend for their transactions.

| **Scheme** | **Privacy** | **Transaction Size** | **Primitive** | **Adoption** | **Trusted Setup** | **Reg. Compliance** |
|---|---|---|---|---|---|---|
| Omniring | $k$-anonymity | $\mathcal{O}(k)$ | Ring signature | - | ✗ | Auditable |
| VCash | Full | $\mathcal{O}(1)$ | Curve Trees | - | ✗ | - |
| Zerocash | Full | $\mathcal{O}(1)$ | zk-SNARKs | Zcash | ✓ | Auditable/Accountable |
| Zerocoin | Full | $\mathcal{O}(1)$ | RSA Accumulator | Zcoin | ✓ | |
| Lelantus | $k$-anonymity | $\mathcal{O}(k)$ | Bulletproof | Firo | ✗ | Auditable |
| CryptoNote | $k$-anonymity | $\mathcal{O}(k)$ | Ring signature | Monero | ✗ | - |
| Mimblewimble | Sender-Receiver Unlink. | $\mathcal{O}(k)$ | Bulletproof | Litecoin | ✗ | - |
| Dash | Sender-Receiver Unlink. | $\mathcal{O}(1)$ | CoinJoin | Dash | ✗ | - |

Table 1: Comparison of UTxO based privacy-preserving schemes. The first two systems are not deployed.

## 4.4 Regulatory Compliance

In this section we will describe some of the techniques that are used to add accountability or auditability to the above described schemes.

Garman et al [25] present an accountable version of the Zerocash protocol. In particular they show how to modify the Zerocash protocol to allow for policies that simultaneously allow the authorities to trace coins as they go from individual to individual and retrieve all of a particular user's transactions and provide an accountable record of when and why those powers were used. They achieve auditability of the coins via a tracing key (viewing key) that is used to encrypt part of the coin. The authority can then decrypt the coins that need to be traced. They also add information to coins such as counters that keep track of the spending limits of the party. Via a zero-knowledge proof the sender needs to prove that the transaction value is below the spending limit or conversely requires a signature from an authority to ensure that the transaction is valid. Thus they also achieve auditability.

Lelantus Spark [39] allows three levels of opt-in visibility into transactions. Incoming view keys allow a designated third party to identify transactions containing outputs destined for an address, as well as the corresponding amounts and encrypted memo data. Full view keys allow a designated third party to additionally identify when received outputs are later spent (but without any recipient data), which enables balance auditing and further enhances accountability in threshold multisignature applications where this property is desired. Payment proofs allow a sender to assert the destination, value, and memo of a coin while proving (in zero knowledge) that it knows the secret data used to produce the coin; this permits more fine-grained disclosure without revealing view keys.

Omniring [42] also presented an extension to their RingCT protocols to achieve trackability and viewability. CryptoNote [76] introduced a feature called trackability (same as auditability), where a user voluntarily delegates a tracking key to a trusted third party, so that the latter can track incoming transactions on behalf of the user. This is particularly useful for a computationally constrained user as tracking incoming transactions requires monitoring all new messages posted on the public ledger. In Omniring, the authors show how the tracking capability can be extended such that the designated third party can also learn the amount to be received in an incoming transaction. They term this as viewability which is useful in scenarios where a user wishes to have incoming transactions to its address audited.

Traceable Monero by Li et al [44] and Accountable Monero by Zhang et al [81] present schemes on how to add auditability and accountability to Monero respectively.

In Table 1 we present an overview of the different UTxO blockchain protocols with native privacy features. We note that we do not include mixing techniques that are explicitly external mechanisms as they are not our main focus. Then, in the rest of this section we discuss deployment challenges and open questions.

### 4.5 Deployment Challenges in Private UTxO based Systems

**Storage costs for validators.** In all UTxO-based systems, excluding Dash and Mimblewimble and other mixing-based approaches, the set of unspent transactions continuously grows. Additionally, the most common technique to prevent double spending of already spent transactions is through the use of nullifiers. These nullifiers must be stored by validators to validate incoming transactions. As a result, the storage costs for validators increase linearly with each transaction posted on the chain.

**Support for light clients.** Light clients are users of a blockchain who operate on devices with limited computing power, such as phones or laptops. Typically, payments in private cryptocurrencies do not require any interaction between the sender and the receiver. This creates the problem of payment notification, meaning the recipient must be informed when they have been paid.

The simplest approach is for the user to scan the blockchain and check every transaction. While this is an issue for non-privacy-preserving transactions as well, in those cases, privacy concerns are minimal, so clients can outsource the search to a third party. However, in privacy-preserving systems, outsourcing compromises the client's privacy. Recent works on oblivious message retrieval [79, 48, 46, 37] address this issue by enabling third parties to obliviously search for transactions and return the results to the recipient without compromising their privacy.

A second challenge for new clients in privacy-preserving blockchains is verifying that they are using the correct version of the blockchain. To do this, they must download the entire chain and verify each transaction. For privacy-preserving systems, this also involves verifying the proofs associated with the transactions. This process can be highly inefficient for clients operating on devices with limited computational resources.

**Open Problem**: Can we devise an anonymous UTxO protocol that helps light clients verify the state of a private blockchain efficiently?

**Need for trusted setup in fully anonymous systems.** Zerocash, the current state-of-the-art, uses zk-SNARKs to achieve full anonymity. These SNARKs require a trusted setup to generate the common reference string. However, Zcash, the real-world implementation of Zerocash, has recently adopted Halo2 [7], a system that eliminates the need for a trusted setup.

## 5 Privacy-preserving transactions in the Account-based Model

We will first present schemes that only achieve confidentiality and then proceed to talk about ones that achieve some notions of anonymity. In Figure 2, we present the landscape of various privacy-preserving mechanisms, categorized along the axes chosen in this SoK. This figure can also assist protocol designers in selecting an appropriate design for privacy-preserving transactions on their blockchain, based on specific constraints and requirements.

### 5.1 Confidentiality Only

The first work in the literature to achieve privacy in account-based cryptocurrencies was that of Zether [8] by Bunz et al. As mentioned in the blueprint for privacy-preserving account-based cryptocurrencies above the scheme requires the state to be maintained as homomorphic objects. In the case of Zether they use homomorphic encryption to maintain these ciphertexts, more specifically they use ElGamal encryption [20]. In Solana [70], the state is encrypted using a twisted Elgamal encryption scheme [45], which is a variant of the Elgamal encryption scheme where a ciphertext is composed of a Pedersen commitment of the encrypted message and a "decryption handle" that binds the encryption randomness with respect to a specific ElGamal public key. This twisted variant aids them in computing their zero-knowledge proofs efficiently.

Very recently, Inco and Circle Research proposed a confidential transaction ERC20 framework [35] which transforms ERC20 tokens into a confidential form that conceals balances and transaction amounts using Fully Homomorphic Encryption (FHE). The main idea is as follows: the sender encrypts their amount of transaction and submits it to the validators, who then homomorphically compare the value of the transaction $(x)$ with the available balance $bal_S$ for that sender. If $x < bal_S$, this comparison returns an encrypted 1, else an encrypted 0. This encrypted value is then passed to a multiplexer that outputs an encryption of $x$ if 1, and an encryption of 0 otherwise. Finally, the validator updates the balance of the sender with the corresponding ciphertext.

### 5.2 Confidentiality and $k$-anonymity

To achieve $k$-anonymity, one approach is to add dummy accounts from the pool of all accounts in the system as described in Sec 3.2.

Zether [8] presents a technique to extend their protocol to achieve an anonymous version of Zether using the blueprint mentioned above. Anonymous Zether [17] proposed by Diamond improves the efficiency of this scheme by presenting a novel many-out-of-many proof that allows a prover to prove knowledge of a certain subset of a fixed list of commitments, as well as that the elements of this subset satisfy certain properties. PriDe CT [32] is a recent work that extends the previous approaches by also allowing for batching of multiple transactions in one big message, i.e., allowing multiple receivers to receive funds in one posted message. They also achieve forward secrecy.

QuisQuis [21] is another cryptocurrency protocol that uses dummy addresses. In their definition of anonymity, the adversary is given a transaction with an anonymity set of size $k$, where the sender

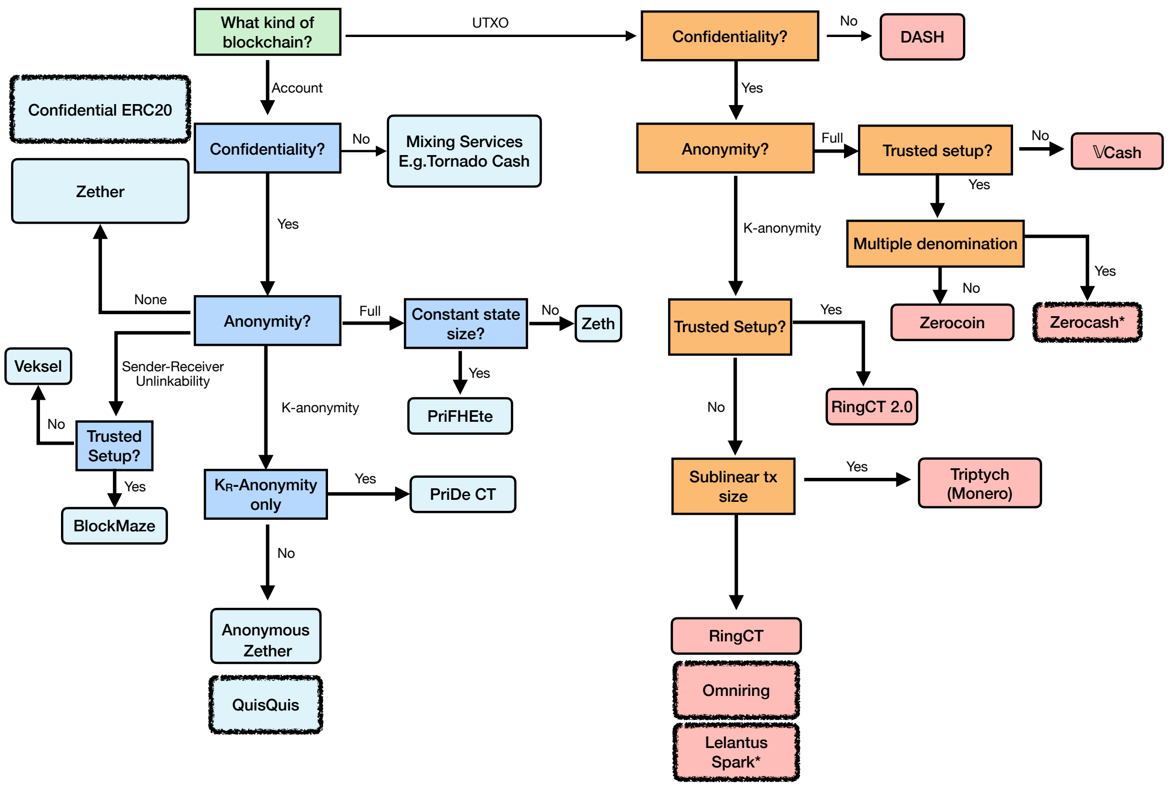


Figure 2: A decision tree presenting an overview of the different privacy-preserving mechanisms. The darker jagged borders represent protocols that have been augmented with auditability or accountability either in the same work or other works. The original Zerocash paper required a trusted setup, but the Zcash deployment of Zerocash is currently moving away from a trusted setup by using Halo2 for their zero-knowledge proofs. Lelantus Spark only had *k*-anonymity, but the deployment Firo, plans other make use of Curve trees to achieve full anonymity.

**Open Problem:** Can we achieve full anonymity and confidentiality efficiently without transaction size being $o(N)$ and validator work being $o(N)$ in the account-based model?

of the transaction is one of two users that are chosen by the adversary. The goal of the adversary is then to determine which of the two users is the sender of the given transaction. Since the sender is hidden within the anonymity set, they only achieve $k$-anonymity. Moreover, we note that QuisQuis is actually a hybrid between UTxO and Account-based because each transaction updates unspent account-balance pairs as in Bitcoin. But they manage to avoid the ever-increasing state issue of the UTxO setting by allowing parties to continuously scan the chain and update their account-balance pairs and deleting their old unspent account-balance pairs via DestroyAcct command. Thus the state of the blockchain only increases with the user base. Finally, QuisQuis suffers from a front-running issue, which does not allow honest transactions to make it to the chain. Campanelli et al [11] present a protocol for homomorphic commitments to key-value maps which hide the key and the value. They use this primitive to extend QuisQuis with a multi-type system, with a state whose size is independent of the number of transactions.

We note that one cannot extend these $k$-anonymity works to achieve full anonymity trivially by setting $k$ to be the size of the total number of parties. This is just not practical since the size of the transaction would be of order of the total number of parties in the system, resulting in a single transaction not fitting in a block hence making this approach impractical. In the next section we describe a protocol that aims to achieve full privacy with constant-sized transactions.

## 5.3 Confidentiality and Full Anonymity

PriFHEte [47] presents a protocol that aims to achieve confidentiality and full anonymity without requiring the transaction size to be of the order of the total number of parties in the system. We note that to achieve full anonymity, the validator who updates the state necessarily needs to touch every account, else some information about the sender/receiver is leaked. Therefore, any account-based protocol that achieves full anonymity with the state represented as (account, balance) pairs will require $O(N)$ work by the validator, where $N$ is the total number of clients in the system. This makes full anonymity with such a data structure for the state infeasible. Therefore it is only a feasibility result that aims to show that transactions need not be the size of the anonymity set to achieve full anonymity. The main idea in PriFHEte is to use FHE to obliviously update each account balance - wherein only the sender and receiver's balances are updated correctly with the value and all other balances are rerandomized by adding an encryption of 0.

## 5.4 Confidentiality and Sender-Receiver Unlinkability

There is another line of work that achieves a notion of sender-receiver unlinkability just as in mixing services described above. In these schemes, the sender sends a coin commitment to a public pool, and the recipient submits a transaction to claim the coin commitment after randomizing it. Thus in these schemes, the receiver must also submit a transaction to update their balance which affects the usability for clients. Blockmaze [31] and Veksel [12] are cryptocurrency examples of such schemes.

**Smart contract-based mixers.** Such protocols achieve sender-receiver unlinkability by implementing a smart-contract based mixer. Möbius [49] is an example of an academic work that designs an Ethereum-based tumbler using stealth addresses and ring signatures. The most well-known example in this category is TornadoCash [14], a deployed Ethereum-based non-custodial cryptocurrency mixer implemented as a smart contract. It operates through smart contracts that collect user deposits, and after depositing funds, a user can later withdraw the same amount to a different address (chosen by the user). Thus, it obfuscates the link between deposit and withdrawal addresses. TornadoCash uses zk-SNARKs in order for a user to prove that they are authorized to withdraw funds without revealing which deposit they initially made.

Trusted execution environments (TEEs) have also been proposed to enhance the confidentiality of smart contract execution and thus the design of L2 mixing services. For example, the Secret Network [54], a standalone Cosmos-based blockchain, uses encrypted inputs, outputs, and state in its secret contracts, which inherently support privacy requirements like sender-receiver unlinkability and Obscuro [57] is an Ethereum L2 rollup protocol designed to achieve data confidentiality and computational privacy by leveraging TEEs.

## 5.5 Emulation of Private UTxOs on Smart Contracts

This line of work aims to integrate Zerocash as a smart contract. Zeth [64] is a proposal for such a scheme. These schemes come with several downsides - 1) The storage costs on account-based cryptocurrencies such as Ethereum is quite expensive, and if a UTxO model is implemented as a smart contract, then all unspent transactions need to be stored on the smart contract. 2) When deployed as a UTxO, the interoperability with other smart contracts is significantly complex 3) Zeth inherits the trusted setup requirement as in Zerocash.

| **Scheme** | **Privacy** | **Primitive** | **Transaction Size** | **Concurrency Issues** | **Regulatory Compliance** |
|---|---|---|---|---|---|
| Zether | Confidentiality only | HE | $\mathcal{O}(1)$ | No | - |
| Confidential ERC20 | Confidentiality only | FHE | $\mathcal{O}(1)$ | No | Auditable |
| Anonymous Zether | Conf. and $k$-anonymity | HE | $\mathcal{O}(k)$ | Yes | - |
| PriDe CT | Conf. and $k_R$-anonymity | HE | $\mathcal{O}(k)$ | Partial | - |
| QuisQuis | Conf. and $k$-anonymity | Commitments | $\mathcal{O}(k)$ | Yes | Auditable/Accountable |
| PriFHEte | Conf. and Full Anonymity | FHE | $\mathcal{O}(1)$ | Yes | - |
| Veksel | Conf. and Sender-Receiver Unlink. | Commitments | $\mathcal{O}(1)$ | No | - |
| Blockmaze | Conf. and Sender-Receiver Unlink. | Commitments | $\mathcal{O}(1)$ | No | - |
| Solana | Confidentiality only | HE | $\mathcal{O}(1)$ | No | - |

Table 2: Comparison of account based privacy-preserving schemes. This table compares different schemes across the privacy they achieve, the main cryptographic primitive used apart from zero-knowledge proofs, the size of the transactions, concurrency issues: which indicate if the protocol needs to add an expensive mechanism to mitigate concurrency issues - a yes implies it is currently inefficient, no implies they dont need any extra mechanism, and partial as in PriDe CT implies they handle with cheaper mechanism. In terms of regulatory compliance, while the presented works may not present an accountable/auditable scheme, we still indicate if there exists other works that augment the given scheme with auditability or accountability. Here $k$ is the anonymity set size, HE stands additive homomorphic encryption, and FHE stands for fully homomorphic encryption. Solana is the only work that is currently deployed.

## 5.6 Regulatory Compliance

In this section, we describe techniques that augment some of the privacy-preserving account-based protocols with auditability.

Papadoulis et al recently proposed AQQUA [58] that builds an auditable payment scheme on top of the QuisQuis protocol. They introduce two new authorities - one for registration and one for auditing. Users first register with the registration authority with real-world credentials and hence provide KYC (Know Your Customer). Their scheme supports the following anti money-laundering policies - restricting the sending/receiving limits in a given time period, restricting the transaction amount limit, allowing a participant to prove that they did not participate in a transaction, and finally allows opening the transaction to a the auditing authority.

Circle [35] introduced the Confidential ERC20 framework, and also include optional viewing and transfer rules to meet regulatory compliance. This is realized by setting some programmable decryption rules at the smart contract level and by allowing an authority to have access to keys that can be used to decrypt transactions or account balances.

We present an overview of the different schemes that implement privacy-preserving account-based protocols in Table 2. As before we only mention cryptocurrency systems with native privacy support and do not mention mixing services in this table.

### 5.7 Deployment Challenges of Privacy-preserving Account-based Systems

**Light Clients.** In QuisQuis [21], clients are expected to delete old accounts to maintain a constant state size. To achieve this, they must scan the blockchain to check if their account has been updated as part of another transaction and issue a destroy account command. However, this is impractical for a light client with limited computational power, as they would need to continuously update their accounts and scan the blockchain. We note that this issue does not arise in other $k$-anonymous account-based protocols, such as Anonymous Zether [17] and PriDe CT [32].

**Front-running.** In a privacy-preserving account-based scheme, each transaction includes a zero-knowledge proof generated about a particular state. For example, when Alice wants to send funds to Charlie, she must compute a zero-knowledge proof to demonstrate that her balance exceeds the transaction amount. This proof is based on her encrypted balance in the state at that time. However, if Bob sends a transaction to Alice that alters the state before her transaction reaches a validator for state update, Alice's proof becomes invalid due to this state change. This issue is termed *front-running* in Zether [8].

To address front-running in the confidentiality-only setting, Zether introduces a solution by holding incoming transactions in a pending state. Time is divided into epochs, each spanning $k$ blocks. At the end of each epoch, these pending transactions are consolidated into the respective accounts. This strategy mitigates front-running, as Alice's proof is now computed solely with respect to her encrypted balance, which is modified only by her outgoing transactions and thus remains fully under her control. A limitation of this approach is that Alice cannot spend incoming funds within the epoch; she must wait until the epoch ends to access those funds. This solution is also applicable in the anonymous case.

**Replay Attacks.** In a replay attack, an attacker resubmits a copy of a legitimate transaction, potentially causing the sender to spend funds twice. In a non-private setting, a common defense is to include a counter in transactions. Validators check that the counters match and update them upon processing each transaction. Since transactions are signed, adversaries cannot forge a signature with an incremented counter. A similar method is employed in confidentiality-only settings. However, this counter-based approach is ineffective in anonymous schemes because transactions involve multiple accounts, only one of which the sender owns. Due to anonymity requirements, the sender's counter cannot be incremented exclusively; instead, counters for all accounts in the transaction must be incremented. This introduces a new type of front-running issue. If Alice includes Bob's account as a dummy, and Bob concurrently submits a transaction with the same counter value, only one transaction will be validated.

The current solution proposed in [8, 17, 47] is to restrict parties to sending only one transaction per epoch. Thereafter, a replayed transaction is simply dicarded. To enforce this restriction, each epoch is associated with a base $g_{epoch}$, derived by hashing a fixed string and the epoch number. Each transaction includes $g_{epoch}^{sk}$, with the sender proving knowledge of $sk$. While this approach ensures replay protection, it restricts parties to a single transaction per epoch.

**Double-spending.** In confidentiality-only schemes, the sender's balance is immediately reduced upon submitting a transaction. However, in anonymous settings with multiple accounts per transaction, updating all account balances encounters the same front-running issue. To circumvent this, balances remain unchanged throughout an epoch, and all parties compute proofs based on the state

| Blockchain | Privacy Options | Ease of Use | Transaction Speed | Unique Feature | Primary Focus |
|---|---|---|---|---|---|
| Zcash | Optional | High | Moderate | Unified Addresses | Privacy Flexibility |
| Monero | Default | High | Moderate | Embedded Privacy | Strong Anonymity |
| Tornado Cash | Default | Moderate | Depends on Ethereum | Ethereum Integration | Ethereum Users |
| Firo | Optional | High | High | Instant Finality | Privacy Simplicity |
| Dash | Optional | High | Very High | InstantSend | Payments |

Table 3: Comparison of user experience across privacy-focused blockchains

at the epoch's start. However, this creates a new vulnerability: *double-spending*. A user could generate two transactions within an epoch and send them to different recipients. Since both transactions are valid concerning the initial epoch state and are anonymous, double-spending is possible. Fortunately, the solution for replay protection –limiting each party to one transaction per epoch as in [8, 17, 47] – also prevents double-spending, as it restricts any party from speaking twice in the same epoch.

**Open Problem**: Can we devise mechanisms to prevent replay attacks, double-spending, and front-running in sender and receiver anonymous account-based payment schemes without limiting parties to a single transaction per epoch?

**Gas fees.** Typical deployment strategies for privacy-preserving transactions in the account-based setting often involve a smart contract, as proposed in Zether [8] and Anonymous Zether [17]. These smart contracts enable users to create private accounts and make payments between private accounts registered under the same contract. While this achieves a certain level of privacy, a notable issue arises: the gas fees required to execute the smart contract must be paid from an externally owned account (EOA), which is a public account. This requirement trivially deanonymizes the sender of the transaction. PriDe CT [32] sidesteps this issue by focusing exclusively on receiver anonymity.

A common approach to preserving sender anonymity is to rely on a relayer that submits the transaction on the sender's behalf and pays the gas fees. For instance, blockchain systems with account abstraction often support gas sponsors, which can act as relayers. While this provides some privacy, the relayer still learns the sender's identity. A potential improvement is to route the transaction through an anonymous channel while including a private transaction to compensate the relayer (e.g., using anonymous tokens). However, designing a protocol that eliminates the reliance on intermediate nodes, such as relayers, while maintaining privacy remains an open challenge. Notably, this issue does not arise in protocols designed as standalone schemes that do not depend on EOAs for gas payments.

**Open Problem**: Can we construct mechanisms to hide gas payments without the need for intermediate nodes?

**Full Anonymity in Account-based Systems.** When the state of the blockchain is a list of account-encrypted balance pairs (say size $N$), it is necessary that all $N$ accounts are updated to achieve full anonymity. This is just impractical due to the massive gas costs that would result from such a transaction.

**Open Problem**: Can we construct a fully anonymous account-based scheme with state size sublinear in $N$?

## 6 User Experience of Production Blockchains

This section focuses on the product experience of four privacy-centric cryptocurrencies: Zcash, Monero, Firo, and Dash, highlighting usability, transaction processes, and accessibility. Although not a standalone cryptocurrency but a mixing service, we also include Tornado Cash given its popularity. Table 3 shows how these blockchains vary across several usability characteristics.

**Zcash.** Zcash provides users with a seamless experience for both private and public transactions. The wallet ecosystem, including Zecwallet and Zecwallet Lite, offers intuitive interfaces for managing funds and selecting privacy levels. Unified Addresses simplify interaction by removing the need to manage multiple address types. The target block interval for Zcash is 2.5 minutes. Shielded transactions, while secure, require more processing time for additional cryptographic operations. It takes overall 6-7 minutes for assurance of transaction finality through the addition of enough blocks. Zashi wallet is their newest wallet which provides additional fortification against network side-channel attack by using Tor nodes.

**Monero.** Monero emphasizes a consistent and privacy-first user experience. Its wallets, such as MyMonero and Monerujo, cater to a range of users, from beginners to advanced. Transactions are straightforward, with anonymity embedded in the process, requiring no additional user actions. MyMonero provides users the choice of syncing with the full blockchain (Advanced mode), as well as a partial sync (Simple mode). The advanced mode may take hours, whereas the simple mode syncs up in a few minutes. Users also have the ability to generate multiple addresses in the same wallet, giving them additional means unlink their identity.

**Tornado Cash.** Tornado Cash offers a simple interface for users seeking to anonymize Ethereum transactions. The web-based platform guides users through depositing and withdrawing assets, using tools like secret notes for privacy. While users must manage their notes securely, the platform's minimalistic design ensures a low barrier to entry. Gas fees and transaction times depend on the Ethereum network, which can influence the overall experience.

**Firo.** Firo's (formerly Zcoin) experience is built around privacy with emphasis on simplicity. Wallets like Stack Wallet and Electrum Firo are designed for ease of use, catering to both private and standard transactions. The platform emphasizes quick transaction finality, which is aimed at making the user experience efficient and reliable. The wallet interfaces and optional privacy settings ensure that both casual and privacy-focused users feel accommodated. Their latest Campfire wallet, which is a fork of the Stack wallet, brings together an interface showing Spark, Lelantus and public balances, and supports addressbook features. While bootstrapping a new address, it takes about a minute to sync with the blockchain, but subsequently finalizes transfers in order of a few seconds.

**Dash.** Dash provides a polished payment-focused experience. Wallets like Dash Core and mobile apps offer fast, easy-to-navigate interfaces with optional privacy features like PrivateSend integrated seamlessly. The settings for PrivasteSend also allows the user to configure the number of mixing round to adjust their trade-off between anonymity level and speed. InstantSend ensures transactions are nearly instantaneous, enhancing usability for everyday purchases. Dash's strong merchant adoption, particularly in regions like Venezuela, reflects its accessibility and real-world practicality. On the downside, the Dash wallet needs to sync with the blockchain for minutes in order to bootstrap. Topper is a service for the wallet which lets users deposit using a credit card after undergoing a light form of KYC. Dash also allows users to stake with its crowdnode validators for yields.

## References

[1] Ghada Almashaqbeh and Ravital Solomon. Sok: Privacy-preserving computing in the blockchain era. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 124–139. IEEE, 2022.

[2] Nasser Alsalami and Bingsheng Zhang. Sok: A systematic study of anonymity in cryptocurrencies. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–9, 2019.

[3] Alireza Arbabi, Ardeshir Shojaeinasab, Behnam Bahrak, and Homayoun Najjaran. Mixing solutions in bitcoin and ethereum ecosystems: A review and tutorial, 2023.

[4] Carsten Baum, James Hsin-yu Chiang, Bernardo David, and Tore Kasper Frederiksen. Sok: Privacy-enhancing technologies in finance. *Cryptology ePrint Archive*, 2023.

[5] Daniel Benarroch, Bryan Gillespie, Ying Tong Lai, and Andrew Miller. Sok: Programmable privacy in distributed systems. *Cryptology ePrint Archive*, 2024.

[6] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[7] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*, 2019.

[8] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.

[9] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[10] Vitalik Buterin. Ethereum whitepaper: A next-generation smart contract and decentralized application platform. `https://ethereum.org/en/whitepaper/`, 2013. Accessed: 2024-08-06.

[11] Matteo Campanelli, Felix Engelmann, and Claudio Orlandi. Zero-knowledge for homomorphic key-value commitments with applications to privacy-preserving ledgers. In *International Conference on Security and Cryptography for Networks*, pages 761–784. Springer, 2022.

[12] Matteo Campanelli and Mathias Hall-Andersen. Veksel: simple, efficient, anonymous payments with large anonymity sets from well-studied assumptions. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 652–666, 2022.

[13] Matteo Campanelli, Mathias Hall-Andersen, and Simon Holmgaard Kamp. Curve trees: Practical and transparent {Zero-Knowledge} accumulators. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4391–4408, 2023.

[14] Tornado Cash. Tornado cash, 2019. Accessed: 2024-11-27.

[15] Panagiotis Chatzigiannis and Foteini Baldimtsi. Miniledger: Compact-sized anonymous and auditable distributed payments. In *European Symposium on Research in Computer Security*, pages 407–429. Springer, 2021.

[16] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. SoK: Auditability and accountability in distributed payment systems. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21: 19th International Conference on Applied Cryptography and Network Security, Part II*, volume 12727 of *Lecture Notes in Computer Science*, pages 311–337, Kamakura, Japan, June 21–24, 2021. Springer, Cham, Switzerland.

[17] Benjamin E Diamond. Many-out-of-many proofs and applications to anonymous zether. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1800–1817. IEEE, 2021.

[18] Thaddeus Dryja. Utreexo: A dynamic hash-based accumulator optimized for the bitcoin UTXO set. Cryptology ePrint Archive, Paper 2019/611, 2019.

[19] Electric Coin Co. Zcash: Privacy-protecting digital currency, 2024. Accessed: 2024-11-27.

[20] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[21] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*, pages 649–678. Springer, 2019.

[22] Firo Project. Firo: Privacy-preserving cryptocurrency, 2024. Accessed: 2024-11-27.

[23] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 657–689. Springer, 2019.

[24] Georg Fuchsbauer and Mathias Wolf. Concurrently secure blind schnorr signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 124–160. Springer, 2024.

[25] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 81–98. Springer, 2017.

[26] Daniel Genkin, Dimitrios Papadopoulos, and Charalampos Papamanthou. Privacy in decentralized cryptocurrencies. *Commun. ACM*, 61(6):78–88, May 2018.

[27] Simin Ghesmati, Walid Fdhila, and Edgar Weippl. Sok: How private is bitcoin? classification and evaluation of bitcoin privacy techniques. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22, New York, NY, USA, 2022. Association for Computing Machinery.

[28] Maxwell Gregory. Coinjoin: Bitcoin privacy for the real world. Bitcointalk Forum: https://bitcointalk.org/index.php?topic=279249.0, 2013. Accessed on 2024-11-26.

[29] Maxwell Gregory. Transaction cut-through. Bitcointalk Forum: https://bitcointalk.org/index.php?topic=281848.0, 2013. Accessed on 2024-11-26.

[30] Dash Core Group. Dash: Digital cash you can spend anywhere. https://www.dash.org, 2024. Accessed: 2024-11-28.

[31] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1446–1463, 2020.

[32] Yue Guo, Harish Karthikeyan, Antigoni Polychroniadou, and Chaddy Huussin. Pride ct: Towards public consensus, private transactions, and forward secrecy in decentralized payments. *Cryptology ePrint Archive*, 2023.

[33] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *ISOC Network and Distributed System Security Symposium – NDSS 2017*, San Diego, CA, USA, February 26 – March 1, 2017. The Internet Society.

[34] Lasse Herskind, Panagiota Katsikouli, and Nicola Dragoni. Privacy and cryptocurrencies—a systematic literature review. *IEEE Access*, 8:54044–54059, 2020.

[35] Inco and Circle Research. Confidential erc20 framework, 2024. Accessed: 2024-11-27.

[36] Tom Elvis Jedusor. Mimblewimble. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt, 2016. Accessed: 2024-11-26.

[37] Yanxue Jia, Varun Madathil, and Aniket Kate. Homerun: High-efficiency oblivious message retrieval, unrestricted. *Cryptology ePrint Archive*, 2024.

[38] Aram Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019:373, 2019.

[39] Aram Jivanyan and Aaron Feickert. Lelantus spark: secure and flexible private transactions. In *International Conference on Financial Cryptography and Data Security*, pages 409–447. Springer, 2022.

[40] Harry A. Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum: Scalable, private smart contracts. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018: 27th USENIX Security Symposium*, pages 1353–1370, Baltimore, MD, USA, August 15–17, 2018. USENIX Association.

[41] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press.

[42] Russell WF Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. Omniring: Scaling private payments without trusted setup. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–48, 2019.

[43] Rujia Li, Qin Wang, Qi Wang, David Galindo, and Mark Ryan. Sok: Tee-assisted confidential smart contract. *Proceedings on Privacy Enhancing Technologies*, 2022.

[44] Yannan Li, Guomin Yang, Willy Susilo, Yong Yu, Man Ho Au, and Dongxi Liu. Traceable monero: Anonymous cryptocurrency with enhanced accountability. *IEEE Transactions on Dependable and Secure Computing*, 18(2):679–691, 2019.

[45] Solana Program Library. The twisted elgamal encryption. https://spl.solana.com/assets/files/twisted_elgamal-2115c6b1e6c62a2bb4516891b8ae9ee0.pdf, 2024. Accessed: 2024-10-23.

[46] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In *Annual International Cryptology Conference*, pages 753–783. Springer, 2022.

[47] Varun Madathil and Alessandra Scafuro. Prifhete: Achieving full-privacy in account-based cryptocurrencies is possible. *Cryptology ePrint Archive*, 2023.

[48] Varun Madathil, Alessandra Scafuro, István András Seres, Omer Shlomovits, and Denis Varlakov. Private signaling. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3309–3326, 2022.

[49] Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. In *Proceedings on Privacy Enhancing Technologies*, number 2, pages 105–121, 2018.

[50] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 ACM Internet Measurement Conference (IMC)*, pages 127–140. ACM, 2013. This paper analyzes Bitcoin transactions to understand user behavior and privacy implications.

[51] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press.

[52] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE symposium on security and privacy*, pages 397–411. IEEE, 2013.

[53] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2024-08-06.

[54] Secret Network. The secret network, 2021. Accessed: 2024-11-27.

[55] Sarang Noether and Brandon Goodell. Triptych: logarithmic-sized linkable ring signatures with applications. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers 15*, pages 337–354. Springer, 2020.

[56] Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[57] Obscuro. Obscuro: Confidential smart contracts for ethereum, 2021. Accessed: 2024-11-27.

[58] George Papadoulis, Danai Balla, Panagiotis Grontas, and Aris Pagourtzis. Aqqua: Augmenting quisquis with auditability. *Cryptology ePrint Archive*, 2024.

[59] Charalampos Papamanthou, Shravan Srinivasan, Nicolas Gailly, Ismael Hishon-Rezaizadeh, Andrus Salumets, and Stjepan Golemac. Reckle trees: Updatable merkle batch proofs with applications. *IACR Cryptol. ePrint Arch.*, page 493, 2024.

[60] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany.

[61] Li Peng, Wei Feng, Zheng Yan, Yafeng Li, Xiaokang Zhou, and Shohei Shimizu. Privacy preservation in permissionless blockchain: A survey. *Digital Communications and Networks*, 7(3):295–307, 2021.

[62] Monero Project. Monero: A private, secure, and untraceable cryptocurrency, 2014. Accessed: 2024-11-27.

[63] Reuben. Curve trees: Global anonymity sets for lelantus spark, March 2024. Accessed: 2024-11-27.

[64] Antoine Rondelet and Michal Zajac. Zeth: On integrating zerocash on ethereum. *arXiv preprint arXiv:1904.00905*, 2019.

[65] Tim Ruffing and Pedro Moreno-Sanchez. Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 133–154. Springer, 2017.

[66] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 345–364. Springer, 2014.

[67] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. P2P mixing and unlinkable bitcoin transactions. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[68] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[69] Savin and contributors on the Bitcoin Development Mailing List. Bitcoin development discussion. https://gnusha.org/pi/bitcoindev/20140106120338.GA14918@savin/, 2014. Accessed: 2024-11-25.

[70] Solana Program Library. zk-token protocol paper, 2023.

[71] Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*, pages 456–474. Springer, 2017.

[72] Xiaoqiang Sun, F. Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE Network*, 35(4):198–205, 2021.

[73] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A 2 l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE symposium on security and privacy (SP)*, pages 1834–1851. IEEE, 2021.

[74] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A$^2$L: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy*, pages 1834–1851, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press.

[75] Aztec Protocol Team. Diversified and stealth keys: Example usage. https://docs.aztec.network/protocol-specs/addresses-and-keys/example-usage/diversified-and-stealth-keys, 2024. Accessed: 2024-11-25.

[76] Nicolas Van Saberhagen. Cryptonote v 2.0. 2013.

[77] Toni Wahrstätter, Matt Solomon, Ben DiFrancesco, and Vitalik Buterin. ERC-5564: Stealth Addresses. *Ethereum Improvement Proposals*, (5564), Aug. 2022. [Online serial].

[78] Dan Wang, Jindong Zhao, and Yingjie Wang. A survey on privacy protection of blockchain: The technology and application. *IEEE Access*, 8:108766–108781, 2020.

[79] Karl Wüst, Sinisa Matetic, Moritz Schneider, Ian Miers, Kari Kostiainen, and Srdjan Čapkun. Zlite: Lightweight clients for shielded zcash transactions using trusted execution. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 179–198. Springer, 2019.

[80] Zcash Community. Zcash improvement proposals (zips), 2024. Accessed: 2024-11-27.

[81] YF Zhang and HX Xu. Accountable monero system with privacy protection. security and communication networks, 2022, article id: 7746341, 2022.