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

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 by enabling secure, decentralized systems for financial transactions, identity management, and beyond. However, its inherent transparency creates serious privacy concerns. Unlike traditional financial systems, where only trusted intermediaries access transaction details, public blockchains expose all data—including pseudonymous identities of senders and receivers and transaction amounts—on an immutable ledger accessible to anyone. This exposure creates vulnerabilities, such as the risk of surveillance, profiling, and exploitation by malicious actors (e.g., MEV [1] attacks).

Achieving privacy for blockchain transactions is complex, requiring a balance between efficiency, and privacy guarantees. Over the past decade, numerous privacy-preserving mechanisms have emerged, all aiming to hide transaction details while maintaining security of the blockchain. However, these solutions

[1] Maximal Extractable Value - the extra profit a blockchain validator, miner, or sequencer can capture by reordering, inserting, or censoring transactions within the blocks they produce.

differ significantly in how they trade off privacy, efficiency, and compatibility with existing systems.

This results in a fragmented landscape of privacy-preserving techniques, each optimized for specific trade-offs and use cases. While some emphasize strong privacy, others prioritize efficiency or simplicity. 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 systematize the techniques and current status of confidential transactions in blockchain systems. Although the literature on privacy-preserving blockchains is extensive, it generally lacks a unified framework to compare the various definitions, techniques, and solutions. Our work fills this gap by providing a structured analysis, identifying key trends, challenges, and open problems, and offering a blueprint for future designs tailored to each system's needs and properties.

First, in Section 2.2, we systematize the various privacy definitions in the literature, including indistinguishability game-based definitions and simulation-based definitions in the Universal Composability (UC) model. These vary widely in scope and rigor, leading to a fragmented understanding of privacy goals. We classify them into three primary categories that capture most systems' objectives and separately define unlinkability, a property relevant to mixing techniques.

Next, in Section 3, we present the main cryptographic techniques used to achieve confidential transactions. We distinguish between techniques for UTxO-based and account-based blockchains, as their underlying data models influence privacy design and implementation. We also briefly discuss transaction mixing techniques as standalone tools and consider mechanisms for regulatory compliance. However, our primary focus is on how mixing techniques are integrated as core functionalities in cryptocurrencies, not external enhancements.

In Sections 4 and 5, we survey the state-of-the-art solutions, including both real-world systems and academic proposals. Our analysis compares solutions by key properties such as privacy level, transaction size, regulatory compliance, and adoption. Figure 2 provides a decision-making graph to guide researchers and developers in selecting the best-suited techniques for their needs.

Finally, we identify **open challenges** and highlight areas for future research.

In Section 7, we also evaluate the usability of popular systems, offering insights into the practical deployment and interaction with privacy-preserving blockchain technologies.

1.2 Scope

In this work, we focus exclusively on privacy-preserving techniques for transactions in permissionless blockchain systems. We focus on permissionless transaction privacy, excluding permissioned ledgers, general private computation (e.g.,

Hawk [47], Arbitrum [46]) and standalone L2 mixers While we briefly explain the cryptographic building blocks used in these systems, such as zero-knowledge proofs, we do not provide a comparative analysis of their techniques or implementations. Similarly, although we discuss key techniques in external or standalone mixing mechanisms as Layer 2 (L2) protocols, we do not extensively compare these services, as that would merit a dedicated study. Instead, our focus is on mixing approaches that inform or contribute to privacy-preserving designs at the cryptocurrency level (Layer 1). The same focus applies to systems supporting accountability and auditability.

1.3 Related Work

The topic of privacy in blockchains has been extensively studied, underscoring its importance and the need for comprehensive SoKs. However, no existing work provides a systematic analysis of the current state of cryptocurrencies with native privacy, the techniques they use, or a practical blueprint to help practitioners choose schemes based on privacy levels and other characteristics. Additionally, no prior SoK has explored the usability of existing systems.

A recent and relevant work is [5], which systematizes cryptographic building blocks like ZK proof systems and addresses broader blockchain privacy topics beyond transactions—areas outside our scope. Similarly, [78] systematizes ZKPs in the blockchain context. Other works with overlapping scope include [40] and [6]; the former explains why traditional ledgers lack anonymity but doesn't cover or distinguish privacy techniques for UTxO and account-based systems, while the latter compares privacy levels but does not cover aspects like efficiency, scalability, and compliance. Works like [67] and [84] also fail to differentiate privacy levels.

The scope of [32] aligns more closely with ours, but it's outdated, less technical, and lacks depth on contemporary systems. Other works focus on narrower domains such as Bitcoin privacy [33], smart contract privacy [49, 9], or DeFi privacy [8]. As noted in Section 1.2, we do not provide an in-depth comparison of mixing services or accountability techniques, instead referring readers to existing SoKs in these areas: accountability [21] and mixing [7].

2 Background

We recall UTxO and Account-based blockchains, and then describe the notions of privacy that are applicable to them. The most common cryptographic primitives used to enhance privacy of transactions are homomorphic encryption schemes, commitments, zero-knowledge proofs and accumulators. We present details of these primitives in Appendix A.

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 mech-

anism, 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. In a "unspent-transaction-output" ledger, the state is a set of unspent coins called UTXOs. Each transaction explicitly spends one or more UTXOs by providing the unlocking credentials (normally a signature over the transaction that verifies under an address associated with the transaction) and simultaneously creates one or more fresh UTXOs specifying new addresses. Because every coin has a unique lineage back to the genesis block, validation is purely local: check that every input UTXO is unspent, correctly authorized, and that the summed input value equals the summed output value plus any fee. There is no concept of a mutable account balance – ownership transfers take the form of consuming old outputs and pouring into brand-new ones. Common examples of cryptocurrencies that are in the UTxO model are Bitcoin [59], Zerocash [74], Dogecoin [2], Cardano [1] etc.

Account-based Model. Here the global state is a key–value map from account addresses to mutable data (balance, nonce, code, storage). A transaction simply states, "debit X coins from account A and credit them to account B", signed by A. Consensus nodes verify that A's balance and nonce are sufficient, then apply the state transition atomically: reduce A's balance, increment A's nonce (to prevent replay), and increase B's balance. Smart-contract accounts execute code during this process, potentially touching many other accounts, but the high-level abstraction is that balances are updated in place rather than re-materialised as new outputs. Common examples of cryptocurrencies that are in the account-based model are Ethereum [15], Sui [3], Solana [76], XRP [4], etc.

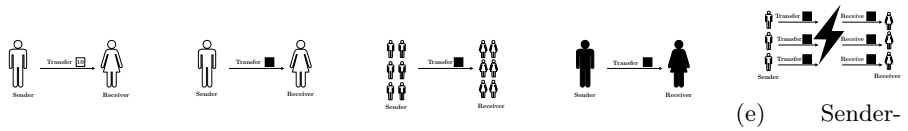
2.2 Notions of Privacy

In this section, we systematize different privacy notions for blockchain transactions. Let $T = (\mathsf{S}, \mathsf{R}, \mathsf{x})$ denote a transaction where S is the Sender, R the Receiver, and x the transaction amount. A transaction without privacy support reveals all three. Notably, revealing the Sender/Receiver typically means exposing their address, not their real identity—a concept known as pseudoanonimity. Given extensive research on the limitations of pseudoanonimity [56], we do not consider it to provide any meaningful level of privacy.

We proceed with defining different privacy flavors starting with the weakest[2].

– **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*).

[2] 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.



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

Fig. 1: Representation of different privacy levels

Confidentiality alone is a relatively weak privacy notion, so stronger concepts have emerged to also hide sender and receiver identities. However, there's no universally accepted definition of privacy that captures all properties of the various schemes. Different works use indistinguishability-based or simulation-based definitions (the latter generally being stronger). The term ledger *indistinguishability* is often used to describe "strong" privacy, but it varies in meaning across the literature depending on the protocol. Here, we clarify these differences by distinguishing between k-anonymity and full anonymity as key privacy properties, noting that the techniques for achieving 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 possible senders or receivers. More formally, for a sender S, the anonymity set is a group of k users such that, for any adversary $\mathcal{A}$, the probability of correctly identifying the true sender is at most $1/k$. The same applies to the receiver. If both the sender and receiver sets have size k, we say the scheme achieves k-anonymity. Otherwise, it achieves k_S-anonymity or k_R-anonymity for sender or receiver side anonymity, respectively.
- **Full Anonymity:** A blockchain transaction provides full anonymity if the 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, making any given transaction indistinguishable from others. An adversary $\mathcal{A}$ cannot, with more than negligible probability, identify the sender or receiver or link transactions involving the same sender.
  In some works, such as in ZCash [74], this property is captured by an indistinguishability game, where the adversary cannot distinguish between two ledger versions that differ in at least one transaction. Assuming there are n distinct addresses, full anonymity can be seen as k-anonymity for $k = n$.

The previous privacy notions typically refer to the level of privacy of a 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 using interaction multi-graphs, as done in [39, 80].

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 cannot, with more than negligible probability, determine whether any two transactions T_i and T_j involve the same sender-receiver pair.

Interaction multi-graphs map payments from senders to receivers, with edges labeled by transaction epochs. Sender-receiver unlinkability requires that all compatible interaction graphs (i.e. those consistent with the observable data) are equally likely. This defines the anonymity set by the number of such graphs, ensuring that even if sender and receiver identities are 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 similarities with k-anonymity, though they are distinct. Sender-receiver unlinkability protects the *relationship* between 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 uses an explicitly defined set (e.g., $k = 10$), sender-receiver unlinkability derives its anonymity set implicitly from the number of compatible interaction graphs, depending on the transaction network's complexity rather than a fixed value.

2.3 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 such as auditability and accountability as defined in [21], 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. Following this, we discuss two general techniques to achieve notions of privacy - namely stealth addresses and mixing techniques.

3.1 Privacy-preserving Account-based Cryptocurrencies

The state. The balance corresonding to accounts 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.

3.2 Privacy-preserving UTxO based Cryptocurrencies

The State. UTxO coins are replaced by 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.

3.3 Stealth Addresses

Stealth addresses were first introduced by Peter Todd [75] 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), where $pk_R = g^{sk_R}$. 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 [83], Bitcoin [75], the Aztec Network [81], Monero [68] and many other cryptocurrencies.

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 [34][3], without relying on intermediaries. Such schemes, typically use 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, i.e. 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 leveraged 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

[3] CoinJoin is a classic decentralized mixing protocol in UTxO-based blockchains that combines inputs and outputs from multiple users into a single transaction and thus achieves sender-receiver unlinkability.

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 [21] 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.

4 Privacy-preserving transactions in the UTxO model

In this section we discuss schemes in the UTxO setting with native privacy features and summarize our overview in Table 1.

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 [57], the first fully anonymous UTxO proposal in the literature. Following the blueprint outlined in Section 3.2, 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 [74] 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 ρ. Looking ahead, this ρ 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 [25] and is a popular cryptocurrency today. The specifications for Zcash are continuously updated and can be found at [86]. A version of Zerocash with accountability was proposed by Garman et al [31].

Curve Trees and $\mathbb{V}$Cash. Campanelli et al [18] 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 [82] 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 [62] 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 [77] 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 [68], a deployed blockchain, launched in 2014, adopted and expanded on the CryptoNote protocol and then the RingCT protocol to enhance user privacy. Triptych [61] 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 [48] 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 [44, 45] presented the Lelantus protocols that are currently implemented as Firo [28] 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 [45] 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 [69] the Lelantus Spark protocol to use curve trees [18] instead of one-out-of-many proofs to achieve full anonymity.

Open Problem 1 *: Can we realize a UTXO cryptocurrency where full nodes store only a constant-size digest, while users create and update fully-anonymous spend proofs with $\mathcal{O}(\log N)$ or better communication as N grows unbounded?*

Challenges:
- **Witness refresh bandwidth & storage**: Even when a Merkle/accumulator membership check is wrapped inside a constant-size SNARK proof, the *private witness* the owner must keep may not be succinct, and also changes every block. Today this means downloading and storing $\Theta(\log N)$ data *per coin per block*.
- **Refresh-time prover overhead**: Each witness change forces the user to recompute the SNARK *plus* a full prover run (memory-heavy, many milliseconds). No design yet cuts *both* communication and proving to $\widetilde{O}(1)$ while avoiding new trust assumptions.
- **Stateless nullifier tracking**: Double-spend prevention still relies on a public set of nullifiers that grows linearly with the number of spends. Achieving the same security without maintaining or downloading that list remains an open problem.

4.3 Confidentiality and Sender-receiver Unlinkability

We now present techniques that achieve sender-receiver unlinkability in the UTxO model. We begin with representative standalone mixing services for the first two categories defined in Section 3.4, then describe Mimblewimble and Dash—cryptocurrencies with native privacy that incorporate mixing techniques.

P2P Mixing. CoinShuffle [72] and its improved version CoinShuffle++ [73] are characteristic P2P mixing protocols. Peers $p_1, p_2, \ldots, p_n$ collaboratively shuffle transaction outputs using pairwise shared keys k_{ij} established via a Non-Interactive Key Exchange (NIKE) protocol. These keys securely encrypt outputs to obfuscate input-output links. After shuffling, all participants sign the final transaction individually. To prevent malicious behavior (e.g., refusing to sign), CoinShuffle++ includes a blame mechanism that identifies and proves misbehavior without revealing sensitive information. An additional variant, ValueShuffle [71], further protects transaction amounts.

Centralized Tumblers. Tumblebit [39] is a representative centralized mixing service. It uses an intermediary, the *Tumbler*, to enable anonymous payments between a sender S and receiver R. The protocol has three phases: *Escrow*, *Puzzle-Solving*, and *Payment*. In the *Escrow Phase*, S locks value v in an address jointly controlled with the Tumbler, while R creates a deposit address. During the *Puzzle-Solving Phase*, the Tumbler sends puzzles Puzzle_i to S, who solves them and submits solutions Sol_i via hash commitments $\mathsf{H}(r)$. Finally, in the *Payment Phase*, R receives the payment, achieving sender-receiver unlinkability. Tumblebit is compatible with Bitcoin and was implemented in the Breeze Wallet by the Stratis blockchain platform. A^2L [79] improves Tumblebit by using adaptor signatures, randomizable puzzles, and better communication complexity.

Mimblewimble. Mimblewimble is deployed in Litecoin and MimblewimbleCoin. First proposed by an anonymous author and Poelstra [42] and later formally analyzed by Fuchsbauer et al. [29], it combines Pedersen commitments for confidential transactions, CoinJoin, and transaction cut-through [35]. Cut-through allows a chain of transactions (e.g., spending txo_1 to create txo_3) to be reduced, mitigating UTxO state growth. However, sender and receiver interaction is required for spending. Fuchsbauer et al. [30] recently proposed non-interactive Mimblewimble transactions that remove this requirement.

Dash. *Dash* [36] is a UTxO-based cryptocurrency with native privacy via CoinJoin. It uses *masternodes* to coordinate mixing, allowing users to specify the number of mixing iterations to enlarge the anonymity set. However, this trust in masternodes (who must not log or collude) limits privacy to sender-receiver unlinkability at best. Unlike fully private cryptocurrencies, Dash's privacy features are optional, letting users choose whether to enable PrivateSend.

4.4 Regulatory Compliance

We now describe techniques used to add accountability or auditability to the above schemes.

Garman et al. [31] present an accountable version of Zerocash, enabling authorities to trace coins as they move and retrieve all transactions of a specific user, with an accountable record of when and why tracing was used. They achieve this via a tracing key that encrypts part of the coin, allowing selective decryption.

Scheme	Privacy	Tx Size	Primitive(s)	Adoption	Trusted Setup	Reg. Compliance
Omniring	k-anonymity	$\mathcal{O}(k)$	Linkable ring signatures Pedersen commitments Bulletproof range proofs	–	✗	Auditable
VCash	Full	$\mathcal{O}(1)$	Curve-Tree accumulator Pedersen-hash commitments Commit&prove Bulletproofs	–	✗	–
Zerocash	Full	$\mathcal{O}(1)$	zk-SNARKs Pedersen commitments CRH Merkle tree PRF-based addresses	Zcash	✓	Auditable/Accountable
Zerocoin	Full	$\mathcal{O}(1)$	Strong-RSA accumulator Pedersen commitments NIZK sig. of knowledge	Zcoin/Firo (legacy)	✓	–
Lelantus	k-anonymity	$\mathcal{O}(k)$	1-out-of-N Sigma proofs Generalized Pedersen com. Bulletproof range proofs	Firo/Beam	✗	Auditable
CryptoNote	k-anonymity	$\mathcal{O}(k)$	MLSAG ring signatures Stealth addresses Pedersen commitments Bulletproof range proofs Key images	Monero	✗	–
Mimblewimble	Sender–receiver unlink.	$\mathcal{O}(k)$	Pedersen commitments Bulletproof range proofs Aggregate Schnorr sigs Cut-through aggregation	Litecoin, Grin	✗	–
Dash	Sender–receiver unlink.	$\mathcal{O}(1)$	CoinJoin mixing Masternode quorum ECDSA signatures	Dash	✗	–

Table 1: Comparison of UTXO-based privacy-preserving schemes with detailed cryptographic primitives. The first two systems are research prototypes.

They also add counters to coins for spending limits, and the sender proves in zero knowledge that transaction values comply with limits or require an authority's signature.

Lelantus Spark [45] provides three levels of opt-in transaction visibility: incoming view keys for output detection, full view keys to track output spending (without recipient data), and payment proofs for fine-grained disclosures.

Omniring [48] extends RingCT to support trackability and viewability. CryptoNote [82] introduced trackability via tracking keys for incoming transactions, while Omniring enhances this by letting the third party learn the amount received, which is useful for auditing.

Traceable Monero [50] and Accountable Monero [87] propose schemes to add auditability and accountability to Monero, respectively.

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 coins 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 who operate on devices with limited computing power, such as phones or laptops. Typically, private

cryptocurrency payments do not require sender-receiver interaction, creating the challenge of payment notification, where the recipient must learn they've been paid.

The simplest approach is for the client to scan the blockchain for relevant transactions. While this is already an issue for non-privacy-preserving transactions, there privacy concerns are minimal, allowing clients to outsource the search to a third party. However, in privacy-preserving systems, outsourcing can compromise client privacy. Recent works on oblivious message retrieval [85, 54, 52, 43] address this by enabling third parties to search for transactions obliviously and return results without compromising privacy.

A second challenge is verifying the blockchain's correctness. New clients must download the chain and verify each transaction, including privacy-preserving proofs. This can be inefficient for clients with limited computational resources.

Open Problem 2 *Can we devise an anonymous UTxO protocol that helps light clients verify the state of a private blockchain efficiently?*

Challenge: To verify for themselves the parties currently are required to download the whole chain. Vurrent efficient solutions require relying on external sources.

Need for trusted setup in fully anonymous systems. Zerocash, a leading protocol, uses zkSNARKs for full anonymity but requires a trusted setup. However, Zcash, its real-world implementation, recently adopted Halo2 [12], which removes the need for a trusted setup.

5 Privacy-preserving transactions in the Account-based Model

We now discuss the account-based setting starting with schemes that only achieve confidentiality and then proceed to talk about ones that achieve some notions of anonymity. We summarize the comparison in Table 2.

5.1 Confidentiality Only

The first work in the literature to achieve privacy in account-based cryptocurrencies was that of Zether [13] 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 [26]. In Solana [76], the state is encrypted using a twisted Elgamal encryption scheme [51], 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 [41] 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 3.1.

Zether [13] presents a technique to extend their protocol to achieve an anonymous version of Zether using the blueprint mentioned above. Anonymous Zether [23] 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 [38] 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 [27] 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 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 [16] 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.

Chow et al. [22] curb state growth by partitioning the address space into fixed-size chunks of k accounts and requiring every ring signature to draw all of

its members from a single chunk. Once a chunk has appeared in k distinct transactions, each account within it must have spent at least once, so the entire chunk can be safely pruned. This bounds the ledger to $O(k \cdot \#\text{unused chunks})$ instead of allowing unbounded expansion. The trade-off is a linear privacy–storage relationship: increasing the ring size k to boost anonymity proportionally increases the worst-case state size.

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 [53] 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 impractical. 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 [37] and Veksel [17] are cryptocurrency examples of such schemes.

Smart contract-based mixers. Such protocols achieve sender-receiver unlinkabibility by implementing a smart-contract based mixer. Möbius [55] 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 [19], a deployed Ethereum-based non-custodial cryptocurrency

Open Problem 3 *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?*

Challenges:
- **Global-state access:** hiding which two accounts change inevitably touches the entire state
- **Succinct oblivious updates:** need an update gadget that changes two encrypted balances while randomising the rest. This is solved only using heavy techniques like FHE.
- **Zero-knowledge cost:** proving correct global updates without a witness of size $\Theta(N)$ is also challenging.

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 [60], 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 [63] 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 [70] 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.

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 [64] 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

Scheme	Privacy	Primitive(s)	Tx Size	Concurrency Issues	Reg. Compliance
Zether	Confidentiality only	ElGamal add. hom. enc. Pedersen commitments Range/Sigma ZK proofs	$\mathcal{O}(1)$	No	–
Confidential ERC20	Confidentiality only	Fully Homomorphic Encryption (FHE) Policy view/transfer keys	$\mathcal{O}(1)$	No	Auditable
Anonymous Zether	Conf. & k-anon.	Twisted ElGamal HE Pedersen commitments Many-out-of-many proofs Range proofs	$\mathcal{O}(k)$	Yes	–
PriDe CT	Conf. & k_R-anon.	Linear hom. enc. Updatable PKE (forward sec.) Pedersen commitments Lightweight ZK proofs	$\mathcal{O}(k)$	Partial	–
QuisQuis	Conf. & k-anon.	Key-hom. (updatable) PKE Homomorphic commitments Bulletproof range proofs NIZK shuffle proofs	$\mathcal{O}(k)$	Yes	Auditable/Accountable
PriFHEte	Conf. & Full anon.	Fully Homomorphic Encryption (FHE) ZK proofs of global update	$\mathcal{O}(1)$	Yes	–
Veksel	Conf. & S–R unlink.	Homomorphic commitments One-many NIZK $\pi_{\text{1-many}}$ (Strong-RSA/DDH)	$\mathcal{O}(1)$	No	–
BlockMaze	Conf. & S–R unlink.	Groth16 zk-SNARKs Pedersen commitments Dual-balance model	$\mathcal{O}(1)$	No	–
Solana	Confidentiality only	Twisted ElGamal HE On-chain ZK proofs (equality, range, validity)	$\mathcal{O}(1)$	No	–

Table 2: Comparison of account-based privacy-preserving schemes with detailed cryptographic primitives. Solana is the only design already deployed on-chain.

the registration authority with real-world credentials and hence provide KYC (Know Your Customer - a regulatory process by which financial or crypto service providers verify and monitor a client's identity and risk profile to prevent money-laundering, terrorism financing, and fraud). 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 [41] 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 [27], 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 [23] and PriDe CT [38].

Concurrency Issues In privacy-preserving account-based schemes, every transaction carries a zero-knowledge proof about a particular global state. Suppose Alice wishes to send funds to Charlie. She first proves using a proof tied to her encrypted balance that her balance exceeds the transfer amount. If, however, Bob's transaction to Alice is processed before hers, the underlying state changes and Alice's proof becomes invalid. Zether [13] refers to this vulnerability as *front-running*.

To mitigate front-running in the confidentiality-only setting, Zether buffers incoming transactions in a *pending* pool. Time is divided into epochs of k blocks; at an epoch's end, pending transactions are batch-applied to the relevant accounts. Alice's proof is therefore computed solely against her balance as modified by her *own* outgoing transactions, which she fully controls. A drawback is that she cannot spend funds received during the current epoch until the next epoch begins. The same buffering idea extends to the anonymous setting.

Replay attacks. In a replay attack, an adversary resubmits a valid transaction, potentially causing the sender to double-spend. In non-private systems a simple counter (nonce) prevents this: each transaction carries the sender's counter value, which validators check and increment. Because signatures bind the counter, an attacker cannot forge a higher value.

The same nonce strategy works for confidentiality-only schemes, but it fails for anonymous schemes. An anonymous transaction touches several accounts, only one of which belongs to the sender, so (to preserve anonymity) validators must increment the counters of *all* involved accounts. This re-introduces a form of front-running: if Alice lists Bob's account as a dummy and Bob simultaneously submits his own transaction with the same counter, only one can succeed.

Current proposals [13, 23, 53] restrict each party to *one* transaction per epoch. Any duplicate is simply discarded. To enforce this, epoch e is associated with a base $g_e := \mathsf{Hash}(\text{“epoch”} \,\|\, e)$. Every transaction includes g_e^{sk}, and the sender proves knowledge of sk. While this guarantees replay protection, it also limits 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 en-

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

Open Problem 4 *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?*

Challenges:
- **Per-account freshness proofs:** counters must advance privately for one real sender yet be verifiable by all.
- **Atomic balance locks:** stopping concurrent spends from overlapping anonymity sets without revealing ownership.
- **Race-free proof generation:** proofs bound to mutable state must stay valid despite interleaved transactions that may update the state.

counters the same front-running issue. To circumvent this, balances remain unchanged throughout an epoch, and all parties compute proofs based on the state 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 [13, 23, 53] – also prevents double-spending, as it restricts any party from speaking twice in the same epoch.

Open Problem 5 *Can we construct mechanisms to hide gas payments without the need for intermediate nodes?*

Challenges:
- **Fee-funding confidentiality:** the chain must verify fee sufficiency while hiding the funding account.
- **Miner-incentive compatibility:** validators must still be sure they will receive the fee in a spendable form.

Gas fees. Typical deployment strategies for privacy-preserving transactions in the account-based setting often involve a smart contract, as proposed in Zether [13] and Anonymous Zether [23]. 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 [38] 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.

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 6 *Can we build an account-based cryptocurrency whose on-chain state grows sub-linearly in the number of users, yet lets users make fully-anonymous, confidential transfers?*

Challenges:
- **Compressed balance commitments:** need an accumulator or vector commitment that supports private in-place edits.
- **Zero-knowledge scalable updates:** updating compressed state must remain succinct and publicly verifiable.
- **Key-image scalability:** detecting double spends without storing one nullifier per account or per transfer.

6 Current Landscape

After surveying state-of-the-art privacy solutions for both UTxO- and account-based systems, we present in Figure 2 a decision tree that synthesizes the landscape of privacy-preserving mechanisms, structured according to the axes defined

in this SoK. This figure further serves as a reference for protocol designers, assisting them in identifying suitable designs for privacy-preserving transactions on their blockchain in light of specific constraints and requirements.

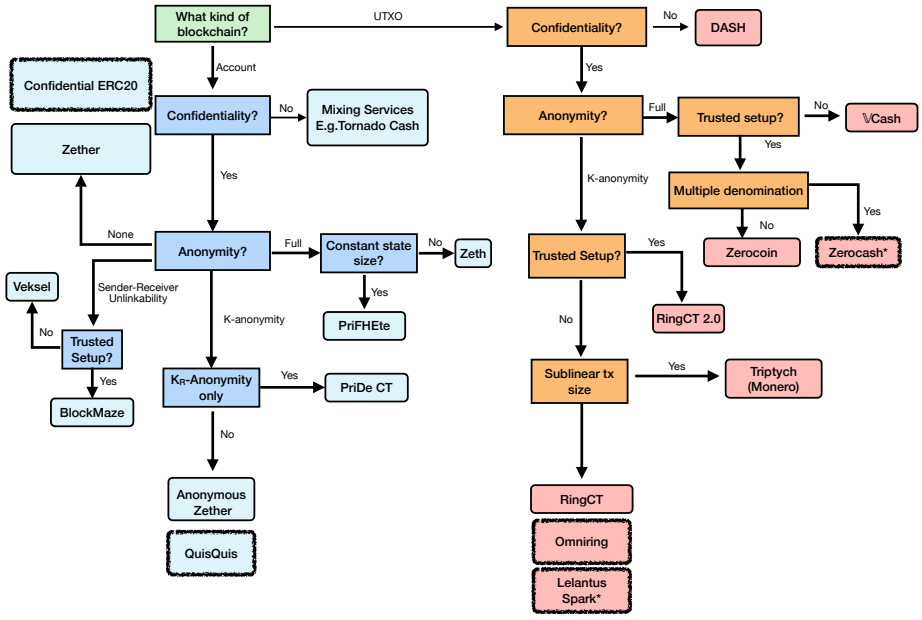


Fig. 2: A decision tree presenting an overview of the different privacy-preserving mechanisms. The darker jagged borders represent protocols that have been augmented with auditability or accountability either in the same work or other works.

7 User Experience of Production Blockchains

We finally review the product experience of four privacy-centric cryptocurrencies – Zcash, Monero, Firo, and Dash – together with the (non-currency) mixer Tornado Cash, focusing on usability, transaction flow, and accessibility. Table 3 summarizes their main characteristics.

Zcash. Zcash supports both transparent and shielded transfers. Wallets such as *Zecwallet* and *Zecwallet Lite* provide intuitive UIs and let users choose their desired privacy level; Unified Addresses eliminate the need to manage multiple address types. The target block time is 2.5 minutes; a shielded transfer typically attains finality after 6–7 minutes (enough confirmations to counter re-orgs)

because of the additional cryptographic work per block. The new *Zashi* wallet routes traffic via Tor to harden against network-layer side-channel attacks.

Monero. Monero offers a privacy-first, uniform workflow. Wallets such as *MyMonero* and *Monerujo* cater to a range of users, from beginners to advanced. Anonymity is automatic – no extra steps are required – yet users may pick between *Simple* (partial) and *Advanced* (full-chain) sync. The former takes a few minutes, the latter several hours. Wallets can also derive multiple sub-addresses, helping users unlink identities across payments.

Tornado Cash. Tornado Cash provides a minimalist web interface for mixing Ethereum assets. Users deposit funds, receive a secret "note," and later withdraw to a fresh address. While note management is manual, the simple workflow keeps the entry barrier low. Gas fees and confirmation latency follow Ethereum network conditions and thus fluctuate with congestion.

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 *PrivateSend* 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.

8 Conclusion

This Systematization of Knowledge reviews over a decade of privacy-preserving payment systems, from *Zerocash* to the latest FHE-based proposals. Our main contributions are a single framework of four privacy notions that applies to both UTXO and account-based designs; a "design map" (see Fig. 2) that visually places major schemes so readers can quickly see which systems achieve relevant privacy and efficiency properties; and six open problems, each paired with the two or three technical challenges that must be cleared to solve it.

Current state. Zero-knowledge proofs have made private transfers practical, but no existing system delivers all three goals at once: *(i)* strong anonymity, *(ii)* on-chain data that grows slower than the number of users, and *(iii)* light-client verification that fits on a resource-constrained device. Work so far divides along three lines: privacy versus scale, support for light clients, and compliance features such as view keys.

Key open problems. We judge three questions as the most urgent: 1) **Open Problem 1** A stateless yet fully anonymous UTXO ledger. 2) **Open Problem 2**: Light clients that can verify the ledger efficiently. 3) **Open Problem 6** — An account-based system whose on-chain state grows sub-linearly while still hiding senders and receivers.

Prospects. *Most near-term progress* is likely on Open Problem 1 and 2: recent vector-commitment and batch-update techniques point the way to compact "delta proofs" for end users. Sean Bowe's recent proposal Tachyon [11] also is a great direction that relies on external server to compute witness updates for state updates recursively. Open Problem 6 remains a *longer-term goal* because it needs a new commitment method that lets balances be updated in place without revealing who owns them and also requires all users to update their witnesses without learning the balance and the indices in the commitment that were updated.

References

1. Cardano website. `https://cardano.org`. Accessed 24 May 2025.
2. Dogecoin website. `https://dogecoin.com`. Accessed 24 May 2025.
3. Sui website. `https://sui.io`. Accessed 24 May 2025.
4. XRP Ledger website. `https://xrpl.org`. Accessed 24 May 2025.
5. 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.
6. 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.
7. Alireza Arbabi, Ardeshir Shojaeinasab, Behnam Bahrak, and Homayoun Najjaran. Mixing solutions in bitcoin and ethereum ecosystems: A review and tutorial, 2023.
8. Carsten Baum, James Hsin-yu Chiang, Bernardo David, and Tore Kasper Frederiksen. Sok: Privacy-enhancing technologies in finance. *Cryptology ePrint Archive*, 2023.
9. Daniel Benarroch, Bryan Gillespie, Ying Tong Lai, and Andrew Miller. Sok: Programmable privacy in distributed systems. *Cryptology ePrint Archive*, 2024.
10. 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.
11. Sean Bowe. Tachyon: Scaling zcash with oblivious synchronization. Accessed 2025-05-28.

12. Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*, 2019.
13. 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.
14. 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.
15. Vitalik Buterin. Ethereum whitepaper: A next-generation smart contract and decentralized application platform. `https://ethereum.org/en/whitepaper/`, 2013. Accessed: 2024-08-06.
16. 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.
17. 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.
18. 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.
19. Tornado Cash. Tornado cash, 2019. Accessed: 2024-11-27.
20. 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.
21. 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.
22. Sherman SM Chow, Christoph Egger, Russell WF Lai, Viktoria Ronge, and Ivy KY Woo. On sustainable ring-based anonymous systems. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 568–583. IEEE, 2023.
23. 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.
24. Thaddeus Dryja. Utreexo: A dynamic hash-based accumulator optimized for the bitcoin UTXO set. Cryptology ePrint Archive, Paper 2019/611, 2019.
25. Electric Coin Co. Zcash: Privacy-protecting digital currency, 2024. Accessed: 2024-11-27.
26. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.
27. 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.

28. Firo Project. Firo: Privacy-preserving cryptocurrency, 2024. Accessed: 2024-11-27.
29. 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.
30. 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.
31. 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.
32. Daniel Genkin, Dimitrios Papadopoulos, and Charalampos Papamanthou. Privacy in decentralized cryptocurrencies. *Commun. ACM*, 61(6):78–88, May 2018.
33. 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.
34. 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.
35. Maxwell Gregory. Transaction cut-through. Bitcointalk Forum: `https://bitcointalk.org/index.php?topic=281848.0`, 2013. Accessed on 2024-11-26.
36. Dash Core Group. Dash: Digital cash you can spend anywhere. `https://www.dash.org`, 2024. Accessed: 2024-11-28.
37. 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.
38. 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.
39. 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.
40. Lasse Herskind, Panagiota Katsikouli, and Nicola Dragoni. Privacy and cryptocurrencies—a systematic literature review. *IEEE Access*, 8:54044–54059, 2020.
41. Inco and Circle Research. Confidential erc20 framework, 2024. Accessed: 2024-11-27.
42. Tom Elvis Jedusor. Mimblewimble. `https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt`, 2016. Accessed: 2024-11-26.
43. Yanxue Jia, Varun Madathil, and Aniket Kate. Homerun: High-efficiency oblivious message retrieval, unrestricted. *Cryptology ePrint Archive*, 2024.
44. Aram Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019:373, 2019.
45. 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.

46. 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.
47. 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.
48. 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.
49. Rujia Li, Qin Wang, Qi Wang, David Galindo, and Mark Ryan. Sok: Tee-assisted confidential smart contract. *Proceedings on Privacy Enhancing Technologies*, 2022.
50. 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.
51. Solana Program Library. The twisted elgamal encryption. `https://spl.solana.com/assets/files/twisted_elgamal-2115c6b1e6c62a2bb4516891b8ae9ee0.pdf`, 2024. Accessed: 2024-10-23.
52. Zeyu Liu and Eran Tromer. Oblivious message retrieval. In *Annual International Cryptology Conference*, pages 753–783. Springer, 2022.
53. Varun Madathil and Alessandra Scafuro. Prifhete: Achieving full-privacy in account-based cryptocurrencies is possible. *Cryptology ePrint Archive*, 2023.
54. 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.
55. Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. In *Proceedings on Privacy Enhancing Technologies*, number 2, pages 105–121, 2018.
56. 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.
57. 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.
58. 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.
59. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008. Accessed: 2024-08-06.
60. Secret Network. The secret network, 2021. Accessed: 2024-11-27.
61. 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.

62. Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.
63. Obscuro. Obscuro: Confidential smart contracts for ethereum, 2021. Accessed: 2024-11-27.
64. George Papadoulis, Danai Balla, Panagiotis Grontas, and Aris Pagourtzis. Aqqua: Augmenting quisquis with auditability. *Cryptology ePrint Archive*, 2024.
65. 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.
66. 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.
67. 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.
68. Monero Project. Monero: A private, secure, and untraceable cryptocurrency, 2014. Accessed: 2024-11-27.
69. Reuben. Curve trees: Global anonymity sets for lelantus spark, March 2024. Accessed: 2024-11-27.
70. Antoine Rondelet and Michal Zajac. Zeth: On integrating zerocash on ethereum. *arXiv preprint arXiv:1904.00905*, 2019.
71. 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.
72. 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.
73. 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.
74. 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.
75. 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.
76. Solana Program Library. zk-token protocol paper, 2023.
77. 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.

78. 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.
79. 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.
80. 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.
81. 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.
82. Nicolas Van Saberhagen. Cryptonote v 2.0. 2013.
83. Toni Wahrstätter, Matt Solomon, Ben DiFrancesco, and Vitalik Buterin. ERC-5564: Stealth Addresses. *Ethereum Improvement Proposals*, (5564), Aug. 2022. [Online serial].
84. Dan Wang, Jindong Zhao, and Yingjie Wang. A survey on privacy protection of blockchain: The technology and application. *IEEE Access*, 8:108766–108781, 2020.
85. 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.
86. Zcash Community. Zcash improvement proposals (zips), 2024. Accessed: 2024-11-27.
87. YF Zhang and HX Xu. Accountable monero system with privacy protection. security and communication networks, 2022, article id: 7746341, 2022.

A 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 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 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 C_m, outputs a bit b that indicates whether the commitment opening was successful.

A commitment scheme should be *hiding*, i.e. 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 [66], 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 π 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 π to the Verifier and the Verifier outputs $0/1$ based on (x, π).

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 Ran-

dom Oracle. Examples of NIZKs used in production include - Bulletproofs [14], or zero-knowledge succinct arguments of knowledge (zk-SNARKs) such as Halo2 [12].

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 acc by running the Add algorithm, while $\mathsf{MemProofCreate}$ creates a membership proof π. 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 [58, 20, 24, 65, 10] 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.