PERMISSIONLESS CRYPTOCURRENCY ABSTRACTION FOR TRANSACTIONS USING NON-NATIVE CRYPTOCURRENCY

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to U.S. Application No. 18/613,542 filed March 22, 2024, the disclosure of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] This specification relates generally to digital asset transactions and more particularly to permissionless cryptocurrency abstraction for transactions using non-native cryptocurrency.

BACKGROUND

[0003] The development of distributed ledger technology has enabled a multitude of digital transactions not available in the pre-Internet world. For example, distributed ledger technology enables users to hold digital assets as stores of value and mediums of exchange with immutability and traceability. In general, a digital asset can be described as a virtual store of value that leverages a distributed ledger to store, record, and validate transactions. An example digital asset includes cryptocurrencies. Distributed ledgers can be referred to as blockchains or chains.

[0004] Each chain network typically maintains a native cryptocurrency, which is used to facilitate transactions in the chain network. For example, the native cryptocurrency is used to account for transaction fees in the chain network. However, native cryptocurrencies are volatile and behave more like a speculative asset rather than a first-class currency. That is, the volatility of native cryptocurrencies makes them a non-reliable store of value. As a result, users have to constantly re-evaluate the cost of transactions with regard the value of the native cryptocurrency with respect to an underlying fiat currency (e.g., United States Dollar (USD)). This creates overhead (in terms of time and consumption of technical resources) for each transaction.

[0005] Further, users interact with multiple chain networks, each chain network having its own native cryptocurrency. For example, a user can use a first cryptocurrency

of a first chain network to account for transaction fees for a transaction in a second chain network. In traditional approaches, the first cryptocurrency has to be converted to a second cryptocurrency (a cryptocurrency that is native to the second chain network). This presents multiple technical issues. For example, conversion of cryptocurrencies itself consumes technical resources (processors, memory, bandwidth). Further, from a user perspective, the user has to initiate conversion transactions, which not only diminishes user experience, but also consumes technical resources on user-side computing devices.

SUMMARY

[0006] This specification describes systems, methods, devices, and other techniques relating to cryptocurrency networks. More particularly, implementations of the present disclosure are directed to permissionless cryptocurrency abstraction for transactions using non-native cryptocurrency. In some implementations, onboarding of user accounts can be executed using a non-native cryptocurrency.

[0007] In general, innovative aspects of the subject matter described in this specification can include actions of creating, by an account factory executed in the chain network, an account for a user within the chain network, the user providing an amount of non-native cryptocurrency to the paymaster to account for a transaction fee for creation of the account in a native cryptocurrency provided from a centralized source of native cryptocurrency, determining, by the paymaster, a maximum transaction fee for creation of the account, the maximum transaction fee being provided in a non-native cryptocurrency, transferring, by the paymaster, the maximum transaction fee from a source of non-native cryptocurrency to the paymaster, after creation of the account, determining, by the paymaster, a total cost of the in the non-native cryptocurrency, and transferring a refund amount from the paymaster to the account based on the total cost, the refund amount being in the non-native cryptocurrency. Other implementations of this aspect include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods, encoded on computer storage devices.

[0008] These and other implementations can each optionally include one or more of the following features: the amount of non-native cryptocurrency is provided from a non-native cryptocurrency source that is provided in the chain network and that is controlled by a provider of the non-native cryptocurrency; the paymaster calls a function of during execution of a paymaster validation to an allowance amount in the non-native cryptocurrency; actions further include providing, from an exchange contract executed within the chain network and to an entrypoint executed within the chain network, a deposit of a native cryptocurrency attributable to a paymaster; actions further include selectively updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint, updating including transferring an exchange amount of non-native

cryptocurrency from the paymaster to a distributed exchange that is executed outside of the chain network, receiving an exchange amount of native cryptocurrency from the distributed exchange, and providing at least a portion of the exchange amount of native cryptocurrency to the entrypoint contract to update the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint; updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint is executed in response to an off-chain worker determining that a balance of the deposit is less than a threshold balance; actions further include, in response to determining, by an off-chain worker, that an exchange rate between the native cryptocurrency and the non-native cryptocurrency has changed by a threshold amount, updating the exchange rate with the paymaster; and the paymaster is provided and deployed to the chain network by an enterprise that provides the non-native cryptocurrency.

[0009] The present disclosure also provides a non-transitory computer-readable storage medium coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with implementations provided herein.

[0010] It is appreciated that the methods and systems in accordance with the present disclosure can include any combination of the aspects and features described herein. That is, methods and systems in accordance with the present disclosure are not limited to the combinations of aspects and features specifically described herein, but also include any combination of the aspects and features provided.

[0011] The details of one or more implementations of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is a conceptual architecture in accordance with implementations of the present disclosure.

[0013] FIGs. 2 and 3 depict example signal flow diagrams in accordance with implementations of the present disclosure.

[0014] FIG. 4 depicts a flowchart of an example process that can be executed in accordance with implementations of the present disclosure.

[0015] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0016] This specification describes systems, methods, devices, and other techniques relating to cryptocurrency networks. More particularly, implementations of the present disclosure are directed to permissionless cryptocurrency abstraction for transactions using non-native cryptocurrency. In some implementations, onboarding of user accounts can be executed using a non-native cryptocurrency. In some implementations, native cryptocurrency is provided from a centralized liquidity source through exchange of non-native cryptocurrency.

[0017] In some implementations, actions include creating, by an account factory executed in the chain network, an account for a user within the chain network, the user providing an amount of non-native cryptocurrency to the paymaster to account for a transaction fee for creation of the account in a native cryptocurrency provided from a centralized source of native cryptocurrency, determining, by the paymaster, a maximum transaction fee for creation of the account, the maximum transaction fee being provided in a non-native cryptocurrency, transferring, by the paymaster, the maximum transaction fee from a source of non-native cryptocurrency to the paymaster, after creation of the account, determining, by the paymaster, a total cost of the in the non-native cryptocurrency, and transferring a refund amount from the paymaster to the account based on the total cost, the refund amount being in the non-native cryptocurrency.

[0018] To provide context for the subject matter of the present disclosure, and as introduced above, users can hold digital assets as stores of value and mediums of exchange. In general, a digital asset can be described as a virtual store of value that leverages a distributed ledger (e.g., blockchain) to store, record and validate transactions. A distributed ledger can be described as a decentralized database that immutably stores transactions across a peer-to-peer network. Distributed ledgers can be referred to as blockchains, or chains that are maintained in a chain network (e.g., a network of peer-to-peer nodes). Example digital assets can include, without limitation, cryptocurrencies (e.g., stablecoins), non-fungible tokens (NFTs), security tokens, and the like. For purposes of non-limiting illustration, implementations of the present disclosure are described in further detail herein with reference to example cryptocurrencies. It is

contemplated, however, that implementations of the present disclosure can be realized using any appropriate digital assets.

[0019] In some examples, a cryptocurrency can be described as a digital currency that uses cryptography to secure transactions that are recorded in a distributed ledger (e.g., blockchain). In some examples, a stablecoin can be described as a cryptocurrency having a value that is tied (pegged) to the value of a real-world, non-digital asset, such as a fiat currency, a commodity, or a financial instrument. Stablecoins are absent the volatility of other cryptocurrencies and, as such, are a useful medium of exchange. An example stablecoin includes, without limitation, USDC™.

[0020] In general, users hold digital assets on a chain and can manage digital assets using externally owned accounts (EOAs). An EOA is an account that is associated with a public key and a private key pair. A digital wallet can be described as a software application that enables users to access and transact digital assets on chains through one or more EOAs. A digital wallet executes off-chain and stores private keys of each of the one or more EOAs. An EOA can be used to initiate a transaction that is executed using a smart contract account (SCA). The SCA is maintained on-chain and executes transactions triggered by user operations (userOps).

[0021] Implementations of the present disclosure are described in further detail herein with reference to example peer-to-peer networks (also referred to herein as chain networks), each maintaining a respective chain. The example peer-to-peer networks can each be described as an Ethereum-compatible network. Ethereum can be described as a decentralized computing infrastructure that executes a virtual machine, referred to as the Ethereum virtual machine (EVM), to execute transactions on a chain. The EVM can be described as a global singleton and operates as a single-instance computer, globally across nodes of the peer-to-peer network. That is, each node in the network executes a local copy of the EVM to execute transactions on the chain. The chain of the peer-to-peer network records the changing state of the EVM as transactions are processed. While Ethereum is referenced herein for purposes of illustration, it is contemplated that implementations of the present disclosure can be realized using any appropriate peer-to-peer network (e.g., networks that provide on-chain computing; EVM-compatible chains).

Example peer-to-peer networks can include, without limitation, Arbitrum, Avalanche, Base, and Tron, among several others.

[0022] Executing transactions in peer-to-peer networks requires the consumption of technical resources (e.g., processing, memory). In other words, computational effort is expended. In the context of Ethereum, gas refers to a unit of measure of the amount of computational effort required to execute specific operations. To pay for a transaction, a transaction fee, referred to as a gas fee in Ethereum, is determined as the amount of gas used to perform operations of the transaction, multiplied by a cost per unit gas. The transaction fee is paid regardless of whether a transaction succeeds or fails. In Ethereum, transaction fees (gas fees) are paid in Ethereum's native cryptocurrency, which is referred to as ether (ETH).

[0023] In using digital assets as a means of exchange, users may have digital assets on one chain that they would like to use on another chain. For example, a user can seek to execute a transaction on a destination chain, but pay transaction fees for the transaction on the destination chain using a digital asset that is maintained on a source chain. However, each chain network maintains a native cryptocurrency, which is used to facilitate transactions in that chain network. Users interact with multiple chain networks, each chain network having its own native cryptocurrency.

[0024] To highlight this, a non-limiting illustrative example can be considered, which includes Alice transferring a digital asset (e.g., a NFT) to Bob in a chain network. The transfer requires computational effort (in Ethereum, gas) to execute the transaction in the chain network, which requires a transaction fee (in Ethereum, gas fee that is paid in ETH). In this example, Alice has an amount of cryptocurrency (e.g., stablecoin) in a SCA in the chain network that Alice would like to use to pay the transaction fee. However, Alice's cryptocurrency on the source chain is non-native with respect to the chain network.

[0025] Such scenarios present multiple issues. For example, in a traditional approach, multiple transactions would need to be executed to facilitate the transaction on the chain network. With continued reference to the non-limiting example above, Alice would have to obtain a sufficient amount of native cryptocurrency to cover the transaction fees. This,

however, can itself require multiple transactions and commensurate expenditure of technical resources, not to mention additional transaction fees. Further, and in some instances, Alice would have to initiate each of the multiple transactions from respective EOAs. This not only consumes resources on the device Alice uses (user-side computing devices), but also diminishes user experience (UX) (e.g., navigating between multiple EOAs and waiting for one transaction to complete before initiating a next transaction).

[0026] In view of the foregoing, implementations of the present disclosure are directed to permissionless cryptocurrency abstraction for transactions using non-native cryptocurrency. Here, cryptocurrency abstraction refers to enabling users to use a non-native cryptocurrency to account for transaction fees in a native cryptocurrency. As described in further detail herein, implementations of the present disclosure provide a permissionless paymaster in the chain network that enables users to use the non-native cryptocurrency to account for transaction fees payable in the native cryptocurrency for transactions executed within the chain network. In some implementations, amounts of native cryptocurrency are provided within the chain network from a centralized liquidity source that can include an exchange contract and a distributed exchange. In some examples, a deposit of native cryptocurrency attributable to the paymaster is selectively updated by the centralized liquidity source. For example, an exchange amount of non-native cryptocurrency is transferred from the paymaster to the distributed exchange (e.g., executed outside of the chain network) and an exchange amount of native cryptocurrency is received from the distributed exchange.

[0027] FIG. 1 depicts a conceptual architecture 100 in accordance with implementations of the present disclosure. In the example of FIG. 1, the architecture 100 includes an off-chain network 102 and an on-chain network 104. The off-chain network 102 represents components that are executed off-chain (e.g., one or more computing devices that are independent of a chain network) and the on-chain network 104 represents components that are executed on a chain network. In the example of FIG. 1, dashed arrows indicate transfers of cryptocurrency between components.

[0028] In the example of FIG. 1, the off-chain network 102 includes a user wallet 110, an alternative memory pool (alt mempool) 112, one or more off-chain workers 114,

and a bundler 118. In some examples, the user wallet 110 is an off-chain program that stores private keys of a user 140, each private key corresponding to an account (e.g., SCA) of the user 140 on a chain maintained by the on-chain network 104. The bundler 118 listens to user operations added to the alt mempool 112 and packages user operations into bundles. Although a single bundler is depicted, it is contemplated that multiple bundlers can be provided. The bundler 118 sends bundles of user operations as regular transactions to nodes of the on-chain network 104.

[0029] In some examples, the alt mempool 112 serves as off-chain storage for user operations that are to be executed on the chain of the on-chain network 104. At a high-level, the alt mempool 112 can be described as a memory pool that holds pending user operations that have not been picked up for execution by the bundler 118. The alt mempool 112 can be described as an alternative to (is different from) so-called canonical memory pools that are used to hold user operations for respective nodes of the on-chain network 104. For example, alt mempools can include specific rules that are to be observed. An example rule of the alt mempool 112 can include, without limitation, that the alt mempool 112 prevents multiple SCAs from reading and writing to the same nonce. Another example rule of the alt mempool 112 can include, without limitation, a paymaster (discussed below) being included on a whitelist of paymasters. Further details of alternative memory pools are provided in Ethereum Request for Comment (ERC) 4337 (ERC-4337) (also referred to as Ethereum Improvement Proposal (EIP) 4337 (EIP-4337)), which is incorporated herein by reference in the entirety for all purposes.

[0030] Components executed in the on-chain network 104 can be provided as smart contracts. Smart contracts can be described as programs that are executed on-chain and are each provided as a collection of code (functions) and data (state) that resides at a specific address on the chain.

[0031] In the example of FIG. 1, the on-chain network 104 includes an entrypoint 120, an account factory 122, a SCA 124, a paymaster 126, and an exchange contract 128 (also referred to as swapper contract). In some examples, the entrypoint 120 receives a bundle (of user operations) from the bundler 118. In some examples, a user operation can include creating a user account (e.g., SCA) for a user within the on-chain network 104.

This can be described as onboarding the user to the on-chain network 104. For example, if the user 140 does not have an account with the on-chain network, a user operation can be executed to trigger the account factory 122 to create the SCA 124 for the user 140. In some examples, a user operation can represent a transaction that a user is requesting be executed between accounts in the on-chain network 104. For each such user operation in the bundle, the entrypoint 120 performs validation before enabling execution of the transaction.

[0032] In the example of FIG. 1, a decentralized exchange (DEX) 130 is provided. The DEX 130 is a third-party peer-to-peer network that enables exchanges of cryptocurrencies. In some examples, the DEX 130 uses a chain that is executed in another on-chain network (i.e., an on-chain network that is different from the on-chain network 104). In the context of the present disclosure, one or more of the off-chain workers 114, the exchange contract 128, and/or the DEX 130 can be collectively considered a centralized liquidity source. As described in further detail herein, the centralized liquidity source enables a reserve of native cryptocurrency to be made available for transaction fees in the native cryptocurrency.

[0033] In some implementations, the one or more off-chain workers 114 can include an EVM client, a transaction processor, and one or more metrics monitors. In some examples, the EVM client can include application binary interfaces (ABIs) to enable interactions with the paymaster 126 and the exchange contract 128. In some examples, the EVM client can update price and swap call data. In some examples, the transaction processor can execute a task (e.g., scheduled task) to query cryptocurrency prices from an exchange (e.g., the DEX 130) and selectively insert a transaction request to update a foreign exchange rate, discussed in further detail herein. In some examples, the transaction processor can execute a task (e.g., scheduled task) to determine a balance of non-native cryptocurrency of the paymaster 126 and selectively insert a request to update the foreign exchange rate (e.g., if the balance of non-native cryptocurrency exceeds a minimum threshold). In some examples, the one or more metrics monitors can monitor the balance of non-native cryptocurrency of the paymaster 126, a balance of native cryptocurrency of the paymaster 126, a number of user operations processed, an amount

of non-native cryptocurrency collected from users, an amount of native cryptocurrency spent, and the like.

[0034] In accordance with implementations of the present disclosure, the paymaster 126 is a smart contract that is permissionless and facilitates payment of transaction fees in the on-chain network 104 on behalf of users. For example, and as described in further detail herein, the paymaster 126 can accept a non-native (from the perspective of the on-chain network 104) cryptocurrency (e.g., USDC™) and account for transaction fees in a native (from the perspective of the on-chain network 104) cryptocurrency (e.g., ETH) of the on-chain network 104. Implementations of the present disclosure will be described in further detail herein with reference to USDC™ as a non-native cryptocurrency and ETH as a native cryptocurrency from the perspective of the on-chain network 104. It is contemplated, however, that implementations of the present disclosure can be realized using an appropriate non-native cryptocurrency and/or any appropriate native cryptocurrency.

[0035] To account for transaction fees for transactions executed in the on-chain network, native cryptocurrency can be provided from the centralized liquidity source and can, for example, be used to compensate the bundler 118 and the paymaster 126 using the native cryptocurrency. In accordance with implementations of the present disclosure, and as described in further detail herein, the centralized liquidity source is considered centralized in that at least a portion of the centralized liquidity source is provisioned and/or controlled by an individual entity.

[0036] In accordance with implementations of the present disclosure, and as introduced above, the paymaster 126 is deployed to the on-chain network 104 (e.g., a destination chain, on which a transaction for the user 140 is to be executed) to account for transaction fees on behalf of users (e.g., the user 140). As described in further detail herein, the paymaster 126 enables transaction fees to be accounted for in a native cryptocurrency using a non-native cryptocurrency held by the user 140. That is, the user 140 need not hold any native cryptocurrency (e.g., in the SCA 124) or convert their non-native cryptocurrency into native cryptocurrency.

[0037] As described in detail herein, the paymaster 126 is permissionless. This means that any SCA on the on-chain network (e.g., the SCA 124) can use the paymaster for any transaction without first requesting and receiving permission to do so. For example, the paymaster 126 enables the SCA 124 to account for user operations executed in the on-chain network 104 entirely in the non-native cryptocurrency (e.g., USDC™). The paymaster 126 does this by accepting payment from the SCA 124 in the non-native cryptocurrency and paying for the user operations in the native cryptocurrency (e.g., ETH). As also described in further detail herein, the paymaster 126 enables the user 140 to be onboarded to the on-chain network 104 (e.g., creating the SCA 124 for the user 140) only using the non-native cryptocurrency. In the context of ETH, this can be referred to as ETH-less onboarding.

[0038] In further detail, the paymaster 126 is funded with native cryptocurrency liquidity from the centralized liquidity source 128. The paymaster 126 pays for user operations in the native cryptocurrency (e.g., ETH), while charging a fee in the non-native cryptocurrency (e.g., USDC™). This enables the user 140 to pay for their transaction entirely in the non-native cryptocurrency.

[0039] In some implementations, the paymaster 126 includes a deposit with the entrypoint 120. The deposit includes an amount of native cryptocurrency of the paymaster 126 held on the entrypoint 120 for payment of transactions. In some implementations, the paymaster 126 includes a stake with the entrypoint 120. The stake includes an amount of native cryptocurrency of the paymaster 126 held on the entrypoint 120 for payment of other operations (e.g., accessing storage of the entrypoint 120).

[0040] In some implementations, the paymaster 126 is configured with a set of functions that can be called to facilitate execution of transactions. Example functions include a paymaster validation function (validatePaymasterUserOp) and a post operation function (postOp). Other example functions include a deposit function (deposit), a withdraw to function (withdrawTo), a get deposit function (getDeposit), an add stake function (addStake), an unlock stake function (unlockStake), and a withdraw stake function (withdrawStake), each of which is discussed in further detail in EIP-4337.

[0041] In some examples, the paymaster validation function is called by the entrypoint 120 to check whether the paymaster 126 will process the transaction (user operation (UserOp)). For example, the paymaster 126 can determine whether the user requesting the transaction has already approved the paymaster 126 to transfer cryptocurrency on the user's behalf. In some examples, if the provider of the paymaster 126 has control over the SCA implementation, a function can be provided in the SCA 124 that only the paymaster 126 can call during execution of validatePaymasterUserOp. This function would give the paymaster 126 an allowance in the non-native cryptocurrency (e.g., the paymaster 126 is approved to pull USDC™ from the SCA 124 up to an allowance amount). More specifically, this function calls "approve" on a non-native cryptocurrency contract, which is automatically allowed because the SCA 124 (owner of funds) is the one calling the function. In some examples, if the provider of the paymaster 126 does not have control over the SCA implementation, the SCA 124 can include generic logic to provide its own validateUserOp function, which would approve the paymaster 126 for non-native cryptocurrency using a user-provided off-chain signature ("permit"). In some examples, a special alt mempool would be needed for this, because of certain ERC-4337 restrictions.

[0042] In some implementations, the SCA 124, from which the non-native cryptocurrency is to be transferred to the paymaster 126, can include an automatic approval feature or can include an allowance feature. In some examples, the automatic approval feature enables the SCA 124 to automatically approve transfer of an amount of the non-native cryptocurrency as requested by the paymaster 126. In some examples, the allowance can be set to an unlimited allowance (MAX_INT) or can be set to a predefined amount. If the allowance is an unlimited allowance, the amount of non-native cryptocurrency requested from the paymaster 126 is not limited (e.g., only limited to the amount of non-native cryptocurrency available from the SCA 124). If the allowance is set to a predefined amount, the amount of non-native cryptocurrency requested from the paymaster 126 is limited by the predefined amount. For example, requests are approved until the predefined amount is expended or is insufficient. In some examples, users can check the balance of the predefined amount and can increase the predefined amount.

However, an increase in the predefined amount expends computation power and, thus, incurs transaction fees.

[0043] In some examples, the paymaster 126 uses a stored foreign exchange rate (fxRate) to calculate a maximum transaction fee (TF_{MAX}) for the transaction (userOp) in the non-native cryptocurrency and executes a transfer from function (transferFrom) to transfer the amount of TF_{MAX} in the non-native cryptocurrency from the SCA 124 (of the user 140) to the paymaster 126. In some examples, the entrypoint 120 supplies a value maxCost to validatePaymasterUserOp to note the maximum amount of native cryptocurrency the transaction might need. In view of this, the paymaster 126 can take the corresponding value from the SCA 124. In executing the postOp function (at the end of the transaction), the entrypoint 120 supplies a value actualGasUsed to the function. Where this is less than the maxCost, the paymaster 126 can refund the difference back to the SCA 124, as described in detail herein. In some examples, the userOp specifies a callGasLimit and maxFeePerGas. This caps the amount of native cryptocurrency the call can use for execution (callGasLimit) as well as sets the maximum the user is willing to pay per unit of native cryptocurrency (maxFeePerGas).

[0044] It can be noted that, prior to initiating the transaction, the user 140 has already enabled the paymaster 126 to transfer from the SCA 124. That is, for example, the SCA 124 stores an approval, which indicates that the paymaster is approved to transfer from the SCA 124. The paymaster validation function succeeds, if the transfer of the non-native cryptocurrency is successful. Otherwise, the paymaster validation function fails and the transaction is not executed.

[0045] If the paymaster validation function succeeds, and one or more other validations are successful (e.g., the validate user operation function (validateOp()) is successful), the entrypoint 120 calls an execute user operation function (executeOp()) of the SCA 124 to execute the transaction within the on-chain network 104.

[0046] After the transaction is complete, the entrypoint 120 calls the post operation function (postOp) of the paymaster 126. In some examples, the entrypoint 120 informs the paymaster 126 (e.g., through the postOp call) of the actual transaction fee (TF_{ACT}) incurred for execution of the transaction. The paymaster 126 pays the actual transaction

fee (TF_{ACT}). For example, the actual transaction fee (TF_{ACT}) is paid to the bundler 118 by the entrypoint 120 using the deposit of native cryptocurrency of the paymaster 126 that is stored on the entrypoint 120.

[0047] In some examples, the paymaster 126 deducts any fees and refunds any excess to the SCA 124 in the non-native cryptocurrency. For example, the paymaster 126 can determine a total cost (T_{COST}) to the user 140 using the following example relationship:

$$T_{COST} = (1 + S) * FXR * (TF_{ACT} + (NC_{PRICE} * A)) + FF \quad (1)$$

where S is a spread, FXR is the foreign exchange rate (e.g., native cryptocurrency to non-native cryptocurrency), NC_{PRICE} is a current market price of computing power in the native crypto-currency (e.g., gas/ETH), A is additional computing power that is to be charged for, and FF is a flat fee (e.g., in the non-native cryptocurrency). The spread can be described as a percentage-based fee. In some implementations, and as described in further detail herein, being centralized, one or more variables of Equation 1 can be set or modified. For example, the entity that controls the paymaster 126 and the exchange contract 128 can set the spread, the foreign exchange rate, and the fixed fee. In some examples, one or more of the off-chain workers 114 can be operated by the entity to set the spread, the foreign exchange rate, and/or the fixed fee.

[0048] In some implementations, as part of execution of the post operation function, the paymaster 126 determines T_{COST} and a refund amount (T_{REF}) as a difference between TF_{MAX} and T_{COST}. The paymaster 126 transfers the refund amount to the SCA 124 in the non-native cryptocurrency.

[0049] In accordance with implementations of the present disclosure, the centralized liquidity source can selectively refresh an amount of native cryptocurrency that is available to the paymaster 126 for accounting for transaction fees in the on-chain network 104. More particularly, the exchange contract 128 can selectively replenish the deposit of native cryptocurrency for the paymaster 126 on the entrypoint 120. Although, as discussed above, the paymaster 126 and the exchange contract 128 are deployed to the on-chain network 104 and are controlled by the same entity, the paymaster 126 and the exchange contract 128 are separate and distinct smart contracts. In this manner, changes

can be made to the centralized liquidity source (e.g., updates to the exchange contract 128) without also having to change the paymaster 126.

[0050] In some implementations, the exchange contract 128 can execute a swap function (swapNonNativeFromPaymasterForNative) to withdraw non-native cryptocurrency accumulated by the paymaster 126 (e.g., from SCAs of users) from the paymaster 126, and exchange (swap) the non-native cryptocurrency for native cryptocurrency on the DEX 130. That is, for example, the exchange contract sends non-native cryptocurrency to the DEX 130 and receives native cryptocurrency from the DEX 130. In some examples, the exchange contract 128 deposits with native cryptocurrency to the entrypoint 120 on behalf of the paymaster 126.

[0051] In some implementations, an off-chain worker 114 (e.g., a transaction processor) can (periodically) query a current value of the foreign exchange rate (e.g., ETH:USDC™) with the DEX 130 and can selectively update the foreign exchange rate with the paymaster 126 (e.g., insert a transaction request to update the foreign exchange rate). For example, the foreign exchange rate of the paymaster 126 can be at a set value. The off-chain worker can compare the set value to the current value (e.g., provided from the DEX 130). If a difference between the set value and the current value exceeds a threshold (e.g., ± X%), the set value of the paymaster 126 can be updated to the current value. For example, an off-chain worker 114 (e.g., an EVM client) can enable interactions with the paymaster 126 (e.g., through one or more ABIs) to update the foreign exchange rate used by the paymaster 126.

[0052] In some implementations, an off-chain worker 114 (e.g., a metrics monitor) can (periodically) determine a balance of non-native cryptocurrency of the paymaster 126 and/or a balance of native cryptocurrency of the paymaster 126. In some examples, if a balance falls below a threshold balance, an off-chain worker 114 (e.g., a transaction processor) can initiate a transaction to exchange cryptocurrencies through the DEX 130. For example, if a balance of the deposit of native cryptocurrency held on the entrypoint 120 falls below a threshold balance, a transaction to exchange can be initiated. In some examples, if a balance exceeds a threshold balance, an off-chain worker 114 (e.g., a transaction processor) can initiate a transaction to exchange cryptocurrencies through the

DEX 130. For example, if a balance of the non-native cryptocurrency held by the paymaster 126 exceeds a threshold balance, a transaction to exchange can be initiated. In some examples, an off-chain worker 114 (e.g., an EVM client) can enable interactions with the paymaster 126 and the exchange contract 128 (e.g., through one or more ABIs) to exchange cryptocurrencies with the DEX 130.

[0053] FIG. 2 depicts an example signal flow diagram 200 in accordance with implementations of the present disclosure. The example signal flow diagram 200 of FIG. 2 represents onboarding of a user to an on-chain network (e.g., creating a SCA for the user) without requiring the user to hold native cryptocurrency of the on-chain network (e.g., ETH-less onboarding introduced above). The example of FIG. 2 includes the user wallet 110, the alt mempool 112, the bundler 114, the entrypoint 120, the account factory 122, the SCA 124, the paymaster 126 a non-native cryptocurrency (NNC) source 202 (e.g., USDC™ contract), and an entity 204. In this example, a user (e.g., the user 140) requests execution of a transaction with the entity 204 (e.g., another SCA, a distributed application (dAPP)).

[0054] The user wallet 110 submits (210) a user operation (userOp) to the alt mempool 112. The bundler 114 retrieves (212) the user operation from the altmempool 112. The bundler 114 bundles the user operation with other user operations and sends (214) the bundle to the entrypoint 120. In some examples, the user operation includes an initialization code field (initCode) and a paymaster and data field (paymasterAndData). In some examples, if the user operation includes initiating creation of an SCA for the user in the chain network, the initialization code field includes an address of the account factory that is to create the SCA (e.g., the account factory 122). If the user operation does not include initiating creation of an SCA (e.g., a SCA already exists for the user), the initialization code field is set to a pred-determined value (e.g., 0). In some examples, the paymaster and data field includes the address of the paymaster 126 within the on-chain network 104.

[0055] In some examples, and as noted above, the user operation can include creation of the SCA 124. For example, and with reference to FIG. 1, the first time that the user 140 interacts with the on-chain network 104, the user does not have a SCA.

Consequently, the SCA 124 can be created for the user 140 by the account factory 122. This can be indicated by the initialization code field (initCode) of the user operation being set to a non-zero value (e.g., userOp.initCode $\neq 0$). Here, the value of the initialization code can be an address of the account factory 122 for deployment of a SCA on the chain (e.g., on the on-chain network 104). In this scenario, a sub-flow 200a is executed, in which the entrypoint 120 calls (216) a create account function (createAccount) of the account factory 122, which initializes (218) the SCA 124.

[0056] The entrypoint 120 validates (220) the user operation with the SCA 124 (e.g., calls validateOp()) and validates (222) the paymaster 126 (e.g., calls validatePaymasterUserOp()). In response to the validation call (e.g., during the paymaster validation), the paymaster 126 checks (224) an allowance given to the paymaster 126 to transfer from the NNC source 202. In some examples, the NNC source 202 is a source of non-native cryptocurrency (e.g., USDC™ contract) to determine whether there is sufficient allowance to transfer an amount of non-native cryptocurrency (e.g., TF_{MAX}) from the SCA 124 to the paymaster 126. In some examples, if the allowance is less than the amount requested (e.g., TF_{MAX}), a sub-flow 200b is executed, in which the paymaster 126 calls (226) an approve paymaster function (approvePaymaster()) of the SCA 124, and the SCA calls an approve function (approve()) of the NNC source 202 (e.g., USDC™ contract). In some examples, the call includes parameters specifying the paymaster 126 (e.g., address of the paymaster 126 in the on-chain network 104) and an allowance amount (e.g., unlimited allowance (MAX_INT)). Here, the sub-flow 200b can be called when the SCA 124 is newly created, per the sub-flow 200a, and is not yet funded with non-native cryptocurrency. Consequently, the paymaster 126 is able to pull non-native cryptocurrency from the NNC source 202 to enable using only the non-native cryptocurrency in on-boarding the SCA 124 (e.g., ETH-less onboarding). In some examples, instead of the NNC source 202, the non-native cryptocurrency can be provided from an onramp solution. For example, from an exchange that allows cost-free sends of non-native cryptocurrency, from an SCA of another user, or from an existing SCA of the user.

[0057] If the allowance is sufficient or the paymaster 126 has been approved, the paymaster 126 calls (230) a transfer from function of the NNC source 202, which transfers (232) the amount of non-native cryptocurrency from the NNC source 202 to the paymaster 126.

[0058] In some examples, the sub-process 200b need not be performed. For example, after creation of the SCA 124, the user can fund the SCA 124 with non-native cryptocurrency, which the paymaster 126 can pull from (instead of pulling from the NNC source 202), as described in detail herein.

[0059] If the validations are successful, the entrypoint 120 calls (234) an execute user operation function (executeOp()) of the SCA 124, which executes (236) the transaction within the entity 204. After the transaction is complete, the entrypoint 120 calls (238) the post operation function (postOp) of the paymaster 126. In some examples, the entrypoint 120 informs the paymaster 126 (e.g., through the postOp call) of the actual transaction fee (TF_{ACT}) incurred for execution of the transaction. The paymaster 126 pays the actual transaction fee (TF_{ACT}). For example, the actual transaction fee (TF_{ACT}) is paid to the bundler 118 by the entrypoint 120 using the deposit of native cryptocurrency of the paymaster 126 that is stored on the entrypoint 120. As described in further detail herein, the paymaster 126 refunds (240) any excess of the non-native cryptocurrency to the SCA 124 (e.g., T_{REF}).

[0060] FIG. 3 is an example signal flow diagram 300 in accordance with implementations of the present disclosure. The example signal flow diagram 300 of FIG. 3 represents execution of a transaction and use of centralized liquidity to provide native cryptocurrency. The example of FIG. 3 includes the user wallet 110, the altmempool 112, the bundler 114, the entrypoint 120, the SCA 124 (already on-boarded), the paymaster 126, an entity 302, the exchange contract 128, and the DEX 130. In this example, a user (e.g., the user 140) requests execution of a transaction with the entity 302 (e.g., another SCA, a distributed application (dAPP)).

[0061] The user wallet 110 submits (310) a user operation (userOp) to the alt mempool 112. The bundler 114 retrieves (312) the user operation from the altmempool 112. The bundler 114 bundles the user operation with other user operations and sends

(314) the bundle to the entrypoint 120. The entrypoint 120 validates (316) the user operation with the SCA 124 (e.g., calls validateOp()) and validates (318) the paymaster 126 (e.g., calls validatePaymasterUserOp()). In response to the validation call (e.g., during the paymaster validation), the paymaster 126 determines the maximum amount of the transaction fee (TF_{MAX}) and calls (320) a transfer from function (transferFrom()) of the SCA 124, which transfers an amount of non-native cryptocurrency (e.g., TF_{MAX}) to the paymaster 126.

[0062] If the validations are successful, the entrypoint 120 calls (324) an execute user operation function (executeOp()) of the SCA 124, which executes (326) the transaction with the entity 302. After the transaction is complete, the entrypoint 120 calls (330) the post operation function (postOp) of the paymaster 126. In some examples, the entrypoint 120 informs the paymaster 126 (e.g., through the postOp call) of the actual transaction fee (TF_{ACT}) incurred for execution of the transaction. The paymaster 126 pays the actual transaction fee (TF_{ACT}). For example, the actual transaction fee (TF_{ACT}) is paid to the bundler 118 by the entrypoint 120 using the deposit of native cryptocurrency of the paymaster 126 that is stored on the entrypoint 120. As described in further detail herein, the paymaster 126 refunds (330) any excess of the non-native cryptocurrency to the SCA 124 (e.g., T_{REF}).

[0063] In some implementations, a sub-flow 300a is periodically executed to refresh an amount of native cryptocurrency available to the paymaster 126. For example, a balance of the deposit of native cryptocurrency on the entrypoint 120 that is available to the paymaster 126 can be periodically checked (e.g., a scheduled task). If the balance is below a threshold balance, the sub-flow 300a can be executed.

[0064] In the example of FIG. 3, the exchange contract 128 initiates (334) an exchange transaction with the paymaster 126 (e.g., in response to prompting from an off-chain worker 114). The paymaster 126 submits (336) an exchange request with the DEX 130, which includes an amount of non-native cryptocurrency. The DEX 130 exchanges the amount of non-native cryptocurrency for an amount of native cryptocurrency, and transfers (338) the amount of native cryptocurrency to the exchange contract 128. The

exchange contract 128 transfers (340) the amount of native cryptocurrency to the entrypoint 120 to refresh the deposit on the entrypoint 120.

[0065] FIG. 4 depicts a flowchart of an example process 400 that can be executed in accordance with implementations of the present disclosure. For convenience, the process 400 will be described as being performed by a system of one or more computers located in one or more locations. For example, components of the architecture 104 of FIG. 1, appropriately programmed, can perform example process 400. Although the flowchart depicts the various stages of the process 400 occurring in a particular order, certain stages may in some implementations be performed in parallel or in a different order than what is depicted in the example process 400 of FIG. 4.

[0066] A user operation is received (402). For example, and as described in detail herein with reference to FIG. 1, the entrypoint 120 receives a user operation (userOp) from the bundler 114. In some examples, the user operation is one of multiple user operations in a bundle that the bundler 114 provides to the entrypoint. In some examples, the user operation is representative of a request to execute a transaction within the on-chain network 104 (e.g., transfer a digital asset from the SCA 124 to another SCA). Validations are executed (404). For example, and as described in detail herein, the entrypoint 120 calls a validate operation function (validateOp()) of the SCA 124 to validate the user operation and calls a validate paymaster function (validatePaymasterUserOp()) of the paymaster 126 to validate the paymaster as accounting for transaction fees due for execution of the user operation.

[0067] TF_{MAX} is determined (406) and TF_{MAX} is transferred from the SCA (408). For example, and as described in detail herein, in response to the paymaster validation call, the paymaster 126 determines the maximum amount of the transaction fee (TF_{MAX}) and calls (320) a transfer from function (transferFrom()) of the SCA 124, which transfers an amount of non-native cryptocurrency (e.g., TF_{MAX}) to the paymaster 126.

[0068] It is determined whether the validations are successful (410). For example, and as described in detail herein, the entrypoint 120 receives responses from the SCA 124 and the paymaster 126, each response indicating whether the respective validation was successful. If the validations are not successful, the transaction fails (412). If the

validations are successful, the transaction is executed (414). For example, and as described in detail herein, the entrypoint 120 calls an execute user operation function (executeOp()) of the SCA 124, which executes the user operation (e.g., to transfer a digital asset to another entity, such as a SCA). TF_{ACT} is provided (416). For example, and as described in detail herein, after the transaction is complete, the entrypoint 120 calls the post operation function (postOp) of the paymaster 126 and informs the paymaster 126 (e.g., through the postOp call) of the actual transaction fee (TF_{ACT}) incurred for execution of the transaction. The paymaster 126 pays the actual transaction fee (TF_{ACT}). For example, the actual transaction fee (TF_{ACT}) is paid to the bundler 118 by the entrypoint 120 using the deposit of native cryptocurrency of the paymaster 126 that is stored on the entrypoint 120.

[0069] T_{COST} is determined (418) and T_{REF} is determined (420). For example, and as described in detail herein, the total cost (T_{COST}) of the transaction is determined in the non-native cryptocurrency using Equation 1, above, and T_{REF} is determined as a difference between TF_{MAX} and T_{COST}. Non-native cryptocurrency is transferred to the SCA (422). For example, and as described in detail herein, the paymaster 126 refunds (330) any excess of the non-native cryptocurrency to the SCA 124.

[0070] Implementations of the present disclosure provide multiple technical improvements. For example, the paymaster of the present disclosure enables users to execute transactions and account for transaction fees in a non-native cryptocurrency, encompassing even an initial onboarding stage. The paymaster of the present disclosure provides users with a seamless, unrestricted transaction experience on a chain network without the prerequisite of holding any cryptocurrency that is native to the chain network. As such, implementations of the present disclosure overcome limitations of traditional approaches, which are typically permissioned, imposing access restrictions, require off-chain user interactions, and/or lack the ability to onboard users to a chain network without the users already holding the native cryptocurrency (at least for their initial transaction on the chain network). Other example improvements is that the paymaster efficiently and accurately maintains the exchange rate between the native cryptocurrency

and the non-native cryptocurrency, and is designed such that the paymaster mitigates malicious actions (e.g., cannot be griefed/swindled).

[0071] This specification uses the term "configured" in connection with systems and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed thereon software, firmware, hardware, or a combination thereof that, in operation, cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions.

[0072] Implementations of the subject matter and the functional operations described in this specification can be realized in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer programs (i.e., one or more modules of computer program instructions) encoded on a tangible non-transitory storage medium for execution by, or to control the operation of, data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. The program instructions can be encoded on an artificially-generated propagated signal (e.g., a machine-generated electrical, optical, or electromagnetic signal) that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus.

[0073] The term "data processing apparatus" refers to data processing hardware and encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can also be, or further include, special purpose logic circuitry (e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit)). The apparatus can optionally include, in addition to hardware, code

that creates an execution environment for computer programs (e.g., code) that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[0074] A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document) in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub-programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a data communication network.

[0075] The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by special purpose logic circuitry (e.g., a FPGA, an ASIC), or by a combination of special purpose logic circuitry and one or more programmed computers.

[0076] Computers suitable for the execution of a computer program can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special purpose logic circuitry. Generally, a computer will also include, or be operatively coupled to receive data from or

transfer data to, or both, one or more mass storage devices for storing data (e.g., magnetic, magneto-optical disks, or optical disks). However, a computer need not have such devices. Moreover, a computer can be embedded in another device (e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver), or a portable storage device (e.g., a universal serial bus (USB) flash drive) to name just a few.

[0077] Computer-readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices (e.g., EPROM, EEPROM, and flash memory devices), magnetic disks (e.g., internal hard disks or removable disks), magneto-optical disks, and CD-ROM and DVD-ROM disks.

[0078] **[0001]**To provide for interaction with a user, implementations of the subject matter described in this specification can be provisioned on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse, a trackball), by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback (e.g., visual feedback, auditory feedback, tactile feedback); and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's device in response to requests received from the web browser. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device (e.g., a smartphone that is running a messaging application), and receiving responsive messages from the user in return.

[0079] **[0002]**Implementations of the subject matter described in this specification can be realized in a computing system that includes a back-end component (e.g., as a data server) a middleware component (e.g., an application server), and/or a front-end component (e.g., a client computer having a graphical user interface, a web browser, or

an app through which a user can interact with implementations of the subject matter described in this specification, or any combination of one or more such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network (LAN) and a wide area network (WAN) (e.g., the Internet).

[0080] **[0003]** The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other. In some implementations, a server transmits data (e.g., an HTML page) to a user device (e.g., for purposes of displaying data to and receiving user input from a user interacting with the device), which acts as a client. Data generated at the user device (e.g., a result of the user interaction) can be received at the server from the device.

[0081] **[0004]** While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or on the scope of what may be claimed, but rather as descriptions of features that may be specific to particular implementations of particular inventions. Certain features that are described in this specification in the context of separate implementations can also be implemented in combination in a single implementation. Conversely, various features that are described in the context of a single implementation can also be implemented in multiple implementations separately or in any suitable sub-combination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a sub-combination or variation of a sub-combination.

[0082] Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all

illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the implementations described above should not be understood as requiring such separation in all implementations, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0083] Particular implementations of the subject matter have been described. Other implementations are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

CLAIMS

What is claimed is:

1. A computer-implemented method for account creation within a chain network, comprising:

creating, by an account factory executed in the chain network, an account for a user within the chain network, the user providing an amount of non-native cryptocurrency to the paymaster to account for a transaction fee for creation of the account in a native cryptocurrency provided from a centralized source of native cryptocurrency;

determining, by the paymaster, a maximum transaction fee for creation of the account, the maximum transaction fee being provided in a non-native cryptocurrency;

transferring, by the paymaster, the maximum transaction fee from a source of non-native cryptocurrency to the paymaster;

after creation of the account, determining, by the paymaster, a total cost of the in the non-native cryptocurrency; and

transferring a refund amount from the paymaster to the account based on the total cost, the refund amount being in the non-native cryptocurrency.

2. The computer-implemented method of claim 1, wherein the amount of non-native cryptocurrency is provided from a non-native cryptocurrency source that is provided in the chain network and that is controlled by a provider of the non-native cryptocurrency.

3. The computer-implemented method of claim 1, wherein the paymaster calls a function of during execution of a paymaster validation to an allowance amount in the non-native cryptocurrency.

4. The computer-implemented method of claim 1, further comprising providing, from an exchange contract executed within the chain network and to an entrypoint

executed within the chain network, a deposit of a native cryptocurrency attributable to a paymaster.

5. The computer-implemented method of claim 4, further comprising selectively updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint, updating comprising:

transferring an exchange amount of non-native cryptocurrency from the paymaster to a distributed exchange that is executed outside of the chain network;

receiving an exchange amount of native cryptocurrency from the distributed exchange; and

providing at least a portion of the exchange amount of native cryptocurrency to the entrypoint contract to update the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint.

6. The computer-implemented method of claim 5, wherein updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint is executed in response to an off-chain worker determining that a balance of the deposit is less than a threshold balance.

7. The computer-implemented method of claim 5, further comprising, in response to determining, by an off-chain worker, that an exchange rate between the native cryptocurrency and the non-native cryptocurrency has changed by a threshold amount, updating the exchange rate with the paymaster.

8. The computer-implemented method of claim 1, wherein the paymaster is provided and deployed to the chain network by an enterprise that provides the non-native cryptocurrency.

9. A non-transitory computer-readable storage medium coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations for account creation within a chain network, the operations comprising:

creating, by an account factory executed in the chain network, an account for a user within the chain network, the user providing an amount of non-native cryptocurrency to the paymaster to account for a transaction fee for creation of the account in a native cryptocurrency provided from a centralized source of native cryptocurrency;

determining, by the paymaster, a maximum transaction fee for creation of the account, the maximum transaction fee being provided in a non-native cryptocurrency;

transferring, by the paymaster, the maximum transaction fee from a source of non-native cryptocurrency to the paymaster;

after creation of the account, determining, by the paymaster, a total cost of the in the non-native cryptocurrency; and

transferring a refund amount from the paymaster to the account based on the total cost, the refund amount being in the non-native cryptocurrency.

10. The non-transitory computer-readable storage medium of claim 9, wherein the amount of non-native cryptocurrency is provided from a non-native cryptocurrency source that is provided in the chain network and that is controlled by a provider of the non-native cryptocurrency.

11. The non-transitory computer-readable storage medium of claim 9, wherein the paymaster calls a function of during execution of a paymaster validation to an allowance amount in the non-native cryptocurrency.

12. The non-transitory computer-readable storage medium of claim 9, wherein operations further comprise providing, from an exchange contract executed within the

chain network and to an entrypoint executed within the chain network, a deposit of a native cryptocurrency attributable to a paymaster.

13. The non-transitory computer-readable storage medium of claim 12, wherein operations further comprise selectively updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint, updating comprising:

transferring an exchange amount of non-native cryptocurrency from the paymaster to a distributed exchange that is executed outside of the chain network;

receiving an exchange amount of native cryptocurrency from the distributed exchange; and

providing at least a portion of the exchange amount of native cryptocurrency to the entrypoint contract to update the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint.

14. The non-transitory computer-readable storage medium of claim 12, wherein updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint is executed in response to an off-chain worker determining that a balance of the deposit is less than a threshold balance.

15. The non-transitory computer-readable storage medium of claim 12, wherein operations further comprise, in response to determining, by an off-chain worker, that an exchange rate between the native cryptocurrency and the non-native cryptocurrency has changed by a threshold amount, updating the exchange rate with the paymaster.

16. The non-transitory computer-readable storage medium of claim 9, wherein the paymaster is provided and deployed to the chain network by an enterprise that provides the non-native cryptocurrency.

15. A system, comprising:

a computing device; and

a computer-readable storage device coupled to the computing device and having instructions stored thereon which, when executed by the computing device, cause the computing device to perform operations for account creation within a chain network, the operations comprising:

creating, by an account factory executed in the chain network, an account for a user within the chain network, the user providing an amount of non-native cryptocurrency to the paymaster to account for a transaction fee for creation of the account in a native cryptocurrency provided from a centralized source of native cryptocurrency;

determining, by the paymaster, a maximum transaction fee for creation of the account, the maximum transaction fee being provided in a non-native cryptocurrency;

transferring, by the paymaster, the maximum transaction fee from a source of non-native cryptocurrency to the paymaster;

after creation of the account, determining, by the paymaster, a total cost of the in the non-native cryptocurrency; and

transferring a refund amount from the paymaster to the account based on the total cost, the refund amount being in the non-native cryptocurrency.

16. The system of claim 15, wherein the amount of non-native cryptocurrency is provided from a non-native cryptocurrency source that is provided in the chain network and that is controlled by a provider of the non-native cryptocurrency.

17. The system of claim 15, wherein the paymaster calls a function of during execution of a paymaster validation to an allowance amount in the non-native cryptocurrency.

18. The system of claim 15, wherein operations further comprise providing, from an exchange contract executed within the chain network and to an entrypoint executed within the chain network, a deposit of a native cryptocurrency attributable to a paymaster.

19. The system of claim 18, wherein operations further comprise selectively updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint, updating comprising:

transferring an exchange amount of non-native cryptocurrency from the paymaster to a distributed exchange that is executed outside of the chain network;

receiving an exchange amount of native cryptocurrency from the distributed exchange; and

providing at least a portion of the exchange amount of native cryptocurrency to the entrypoint contract to update the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint.

20. The system of claim 18, wherein updating the deposit of the native cryptocurrency attributable to the paymaster within the entrypoint is executed in response to an off-chain worker determining that a balance of the deposit is less than a threshold balance.