Electronic Dollars (eUSD) Whitepaper

MobileCoin

October 12, 2022

Abstract

MobileCoin is a distributed ledger technology and oblivious computing platform that enables payments on mobile devices. In this paper, we introduce a mechanism for creating multiple tokens on our chain which we call "Confidential Tokens." To our knowledge this is the first blockchain that has delivered support for multiple private tokens on layer one. Assets created via this mechanism can be transacted in essentially the same way as MOB and enjoy all of the same privacy guarantees. Additionally, the blockchain and verifiers are blind as to which token type is being transacted. At the same time, minting and burning operations are transparent by design, so that all participants are always in a position to audit the total supply of any such token.

This mechanism has been specifically designed to support the creation of fully-collateralized stablecoins, meaning that users will no longer have to choose between price stability, privacy, and a high-quality user experience. We present mechanisms to support stabilized assets with auditable minting and burning, so the supply is always known and verifiable with the backing asset.

The first sections of this document will describe this general mechanism which could be the basis of many stablecoins. The last section will describe the specifics of how this is applied to create eUSD.

Contents

1 Introduction & Background

The evolution of decentralized finance (DeFi) has included numerous innovations to produce assets of many different types, with varying behaviors depending on the intended use case. One such use case is to represent assets from various contexts in one ecosystem. For example, the ERC20 [ERC] is one of the most well-known smart contracts on Ethereum which empowers the creation of new assets.

1.1 Multiple Tokens on One Chain

Not all blockchains support the creation and manipulation of multiple tokens on their chains. Many chains, like Bitcoin, support only a single asset. Of those that support multiple tokens, the strategies used for this differ from blockchain to blockchain.

In the Ethereum ecosystem, all secondary assets are created using smart contracts, and secondary asset balances are represented using smart contract states. Most of these assets are ERC20 tokens, and this creates a standard interface with which DeFi applications can be compatible.

There are some consequences of this decision. Specifically, transactions involving USDC have gas fees which must be paid in Ethereum. The amount of these fees depends on how efficient the ERC20 contract is[1].

In the Algorand ecosystem, secondary assets are not created using smart contracts. Instead there is an interface for "Algorand Standard Assets" in the core blockchain logic[Alg].

In the Cardano ecosystem, there is similar native support for user-defined assets in the blockchain, and smart contracting functionality is not a prerequisite[Car].

To our knowledge, there are no *private* layer-one blockchains that have shipped support for multiple tokens[2].

1.2 Ecosystem Governance and Fees

In Ethereum, Algorand, and Cardano, the creation of new assets is completely permissionless. Anyone can register a new instance of a smart contract at some address and (for the cost of gas fees) post it to the blockchain. However, there is still a privileged token at the center of the ecosystem (e.g. Ether), which is the only one that can be used to pay transaction fees. In proof of stake systems, this is also typically the token that is staked to secure the network.

Thus, a user who holds USDC may not be able to send their USDC anywhere, unless they obtain some Ether first. This is necessary to prevent denial-of-service attacks on the network – if any user-created token can be minted in any amount and then used to pay gas fees, then transaction fees would no longer be able to effectively rate limit the network.

Despite this necessity, the result is still a negative user experience, since it adds complexity, particularly for first-time users. One example of an attempted work-around is the introduction of "gasless-sends" in the USDC v2.0 smart contract [Haw] – essentially, users may optionally attempt to delegate the payment of gas fees to some other address or smart contract, which may agree in some applications to cover those fees, and so avoid requiring users to buy Ether themselves directly.

An alternative, which to our knowledge has not been explored, is to allow the blockchain to designate certain "trusted" assets as being usable to pay fees directly. For example, the Ethereum foundation (the governance body) could decide that they trust that USDC is unlikely to be debased and become a denial-of-service vector, and so they could propose to modify the network to allow USDC to be used

[1] A major goal in USDC smart contract development is increasing this efficiency to reduce these fees[Haw].
[2] One proposal [AF] for this was generously shared with us during discussion.

for fees. At this point users who hold only USDC would not be required to purchase Ether. One may imagine that this would have dramatic impacts on tokenomics and demand for Ether, but the node operators themselves are free to convert any collected USDC fees to Ether if they so desire, which significantly limits this impact. In essence, such an arrangement moves the burden of trading USDC for Ether from the users to the node operators, which is to the benefit of the end users and improves the user experience.

1.3 Stable Asset Classes

One class of assets which are vital to DeFi are stable assets, often referred to as "stablecoins." The intention of a given stablecoin is for the value of the asset to remain stable relative to some other asset, often fiat currency. Such stablecoins are "pegged" to the target asset, and the exchange rate at which they trade is called the "peg." Some prominent stablecoins are USD Coin [Cir], TrueUSD [Tru], and Pax Dollar [Pax] which are pegged to the dollar.

Some DeFi products build on top of stablecoins, offering features such as appreciation on tokens contributed to liquidity pools [Com], or developing protocols to protect against risk via insured baskets [Res].

We can categorize stable assets by their approach to maintaining the peg:

- Fully-collateralized
- Algorithmic

1.3.1 Fully-collateralized Stable Assets

These are assets whose total supply is backed by a corresponding supply of the target asset, which *fully collateralizes* the peg.

These assets have a *redemption process*, whereby token holders may exchange the stablecoin for its collateral. This reduces the total supply of the stablecoin. This mechanism differs from case to case, and there may be limits on how often it can occur. However, confidence in the redemption process encourages the market to continue to value the stablecoin at or very close to the peg at all times.

(There is also a similar "purchase" process whereby the stablecoin can be purchased in exchange for collateral, and the total supply of the stablecoin is increased.)

The collateral for a stablecoin can be held either on or off-chain. For example, the collateral for the stablecoin USDC is held in bank accounts by the Circle organization [Cir]. To establish confidence in this arrangement, Circle publishes regular attestation reports from 3rd party auditors, and maintains licenses which increase their compliance obligations as a financial institution.

Other stablecoins are backed by on-chain funds. For example, the Wrapped Bitcoin project supports the wBTC token on the Ethereum network[WBT]. This token is meant to represent Bitcoin in a 1:1 manner on the Ethereum network, in order to make use of the Bitcoin currency in Ethereum DeFi applications. wBTC is governed by a DAO ("decentralized autonomous organization"), and the transparent nature of the Bitcoin and Ethereum networks contributes significantly to the ability of third parties to audit the reserves of Bitcoin which collateralize wBTC.

In both cases here, the average user is not expected to directly carry out the purchase or redemption process. In the case of USDC, a Circle account is needed to participate, and in the case of wBTC, the purchase and redemption processes are carried out by "merchants." Instead, the average user simply purchases the corresponding token in a cryptocurrency exchange at any time they like.

1.3.2 Algorithmically Stable Assets

Some stablecoins are not fully collateralized – instead they have a strategy or "algorithm" which is implemented in on-chain logic. This algorithm creates a system of incentives which encourages market participants to value the stablecoin according to the peg.

It is out of scope to try to describe the various strategies in detail, but examples of the mechanisms employed include:

- Issuing and burning the stablecoin
- Trading the stablecoin for collateral (perhaps on a decentralized exchange)

- Offering users either the stablecoin, or a yield, in exchange for depositing collateral.

This practice can be risky, especially in environments with limited or variable liquidity. A recent example of a peg failure was Terra[Vol]. Another approach is to combine aspects of algorithmic and collateral-backed stablecoins. A notable example of this approach is MakerDAO's DAI [Mak], which uses an algorithmically managed basket of cryptocurrency collateral to maintain the peg of DAI to one US dollar.

1.4 Confidentiality and Stablecoins

Today, there are many so-called "Privacy Coin" projects that create blockchains using sophisticated cryptographic techniques to protect the privacy of transactions. These techniques are now quite advanced, and can completely conceal the sender, amount, and recipient of each transaction.

Generally, each such project has its own blockchain, as they typically require deep and fundamental changes to the native code in order to work (and work efficiently).

For projects that don't have native privacy properties, like Bitcoin and Ethereum, instead there has traditionally been a focus on "mixers" which are services where coins can be sent and withdrawn with the goal of breaking the link between sender and recipient.

This is the closest thing that currently exists to confidential stablecoin transactions. However, mixers have several noted drawbacks – the deposits have to come out of the mixer at some point in order to use them, back into the non-private protocol, at which point amounts and addresses are visible. There are also a number of user-experience issues with mixers, such as unspecified amounts of time that the user must wait between deposits and withdrawals. This is because if you immediately deposit your funds and then withdraw the exact amount a short while later to a new address, observers can infer which funds went where and the entire privacy mechanism is defeated.

To our knowledge, no project has created a native stablecoin with privacy properties, which is a first-class citizen in the ecosystem, and which never requires the use of "non-private" transaction technologies to use normally. In short, no one has yet actually created a private digital dollar.

2 MobileCoin Multiple Token Transaction Protocol

2.1 Confidential Token Types

MobileCoin has proposed a way to extend our existing privacy preserving transaction protocol to accommodate the creation of additional tokens which transact on our network.

Our primary goals here are:

- Choosing to transact in one of the tokens offers the same privacy around sender, recipient and amount as traditional MOB transactions
- Choosing to transact in one of the tokens offers the same speed and easy user experience
- Choosing to transact in one of the tokens does not require new account keys or new public addresses. Existing MOB addresses will now seamlessly support balances in tokens such as stablecoins as well.

We set for ourselves an additional goal:

- Transactions using various tokens are indistinguishable on our blockchain – no one will be able to infer what token you are transacting in.

At the same time, auditability is a critical issue for the supply of these new tokens. For MOB, the supply is very simple – a fixed supply of 250,000,000 MOB was created [koe21] in the **Origin Block** when the network was started, and all transactions are balanced, so there is no mechanism to create new MOB, and there are no burn mechanics built into the blockchain.[3]

[3]Each transaction confirms that MOB was not created or destroyed, but a user could choose to burn their MOB, and MOB will be lost in the course of human behavior and management of funds. Therefore, MOB has a fixed supply, and is naturally deflationary.

For new tokens, we want the creator to be able to determine how the supply works and how it changes. Particularly if the token is a fully-collateralized stablecoin, the manager of the reserve must be in a position to regulate the supply of the token so that they match the reserve. This means that they must have the capacity to mint and burn these tokens. To ensure the security of this arrangement, there is by design no privacy around minting and burning operations – all mint and burn operations are auditable in the clear from the ledger. This enables users to verify that the supply of tokens is exactly as it is intended to be and that runaway minting has not occurred. Aside from minting, transactions are always balanced, and so there is no other mechanism by which the supply can increase.

At the protocol level, every token on the blockchain (including MOB) will have a **Token Id**. The Token ID is a 64-bit integer, and every transaction output (TxOut) now has an associated Token ID. Historical TxOut's implicitly have a Token ID of 0 (for MOB), and represent an amount of MOB. TxOut's with other Token IDs represent amounts of a different token.

The MobileCoin blockchain will now support multiple **Token IDs** at the protocol layer. This is an append-only list of 64-bit integers, so that transaction outputs (TxOuts) can always be spent, no matter their token type. If a token type is deprecated, then only the minting of that Token ID is prohibited, and that can be enforced by removing all governors from the `TokensConfig`.

In the rest of this section, we'll describe the specific protocol changes we propose to achieve these goals.

2.1.1 Masked Token IDs and Amounts

Masked Token IDs are a new field which is added to every MobileCoin TxOut, except of course historical TxOut's which are already in the chain before this change.

Every MobileCoin TxOut contains an "Amount" structure which stores a masked representation of value.

The amount structure is as follows:

- Commitment
- Masked Value
- Masked Token Id (new)

The *commitment* is a Pedersen commitment to the numeric amount of the token, represented as an unsigned 64-bit integer in the smallest representable unit.[4] The Pedersen commitment is the sum

$$v * H + v_{blinding} * G \ .$$

Here v is the numeric value, H is the generator curve point used with the value, and G is the Ristretto base point. $v_{blinding}$ is a scalar value derived from the "TxOut shared secret", which comes from a key exchange between the sender and the recipient.

The masked value is that same 64-bit integer v, but this time, it is encrypted in a more conventional way using the TxOut shared secret. In particular, we hash the TxOut shared secret to produce a pseudorandom 64-bit value, and xor this on top of the bits of v. This enables the true recipient to guess the value for a TxOut (given their guess for the TxOut shared secret), and then confirm it by recomputing the commitment based on this result. The consensus validator itself does not validate the masked value fields – it only validates the commitments.

We extend this as follows:

- The generator H that is used for the amount commitment will now be H_i, varying depending on the Token ID. (H_0 will be equal to the generator in use today for MOB.)
- The Token ID is "masked" using a hash value derived from the TxOut shared secret, to produce the *masked_token_id* field, in very much the same way as the *masked_value* (however, there is domain separation so that the masking fields are not reused.)

[4] For MOB this is "picoMOB," and for any subsequent tokens, terminology around this and how it maps to the user-facing amount of currency will be determined on a case-by-case basis.

This ensures that view key scanning can continue working with minimal impact. Given a guess for the TxOut shared secret, a client can then attempt to unmask both the masked value and the masked Token ID, and compute the blinding factor. Then with these assumptions, they can attempt to construct the corresponding commitment – if this matches the TxOut then view key matching has succeeded.

For historical TxOut's that do not have a masked Token ID, clients can infer that the Token ID is zero, and view key matching will then work the same as before, since H_0 did not change.

These changes have minimal impact on proofs of balance and range proofs, since this all works pretty much the same no matter which Pedersen generator we work over. However, it is important that the Pedersen generators H_i are cryptographically orthogonal.

These changes also have minimal impact on Fog, it simply has to pipe through the masked Token ID.

2.1.2 Generators Specific to the Token ID

Previously, we used the same curve base point for all generators used for the Pedersen commitments to the amount in a TxOut. In other words, the same group generator (Ristretto [dHLA16] with a set basepoint), g, was used for the commitment, c, to the amount value, a, as

$$c = g^a$$

In order to support multiple token types, we will now require that a transaction specify the commitment to the amount as well as the Token ID using a generator with a basepoint derived from the Token ID. The base point is derived using a hash-to-curve operation. The construction gives us the property that all the generators for Token IDs are cryptographically "orthogonal," so that it is intractable for anyone to find a linear relationship between them.

For backward compatibility, the MOB generator will be the same as before, the known quantity now designated as H_0. MOB's Token ID will be 0 as well, and is the only special case.

Subsequent Token IDs, i, will derive their generator basepoint, H_i, from the Token ID with the following (rusty pseudocode):

```
// Use the standard Ristretto basepoint as defined by the curve25519 dalek team
use curve25519_dalek::constants::RISTRETTO_BASEPOINT_POINT;

// Best practice to set domain tags for all hashes
let HASH_TO_POINT_DOMAIN_TAG = b"mc_onetime_key_hash_to_point";

// Hash the concatenation of the domain tag with the basepoint-XORed token_id
let mut hasher = Blake2b512::new();
hasher.update(&HASH_TO_POINT_DOMAIN_TAG);
let xored_token_id = RISTRETTO_BASEPOINT ^ token_id;
hasher.update(&xored_token_id);

// Get the Ristretto basepoint from the hash
let basepoint_for_token_id = RistrettoPoint::from_hash(hasher);
```

All of the transaction operations which require generators will use a generator relative to the Token ID. This includes the range proofs in the ring confidential transactions and the ring signature MLSAGs. It is not possible to construct a range proof relative to another generator if those generators are orthogonal. Thus, it is guaranteed that all transaction inputs and outputs are using the same Token IDs. This is due to the homomorphic property of Pedersen Commitments, namely that addition on the commitments preserves the additive relationship of the pre-committed values, as long as they are using the same group generator.

The verifier, therefore, will show that the sum of inputs does not overflow, and that the sum of outputs matches the sum of the inputs, all relative to the H_i, which is revealed to the verifier. For this reason, the calculation of H_i must be constant time, so that a sidechannel observer would also not be able to determine which Token ID was used in the transaction.

2.1.3 Confirming Token ID with Subaddresses

The Token ID mechanism is orthogonal to the subaddress mechanism – subaddresses do not directly affect "view key matching" as described earlier. The derivation of the subaddress, and the one-time private key of a TxOut, are not affected by any of the changes relating to the Pedersen commitment or the masked Token ID. All of this continues to work exactly as before.

2.2 Authorized Minting and Burning of Assets

A normal MobileCoin transfer transaction contains a set of inputs, which are marked spent, to create (mint) a new set of outputs. The network validators confirm that the sum of the value of the inputs is equal to the sum of the value of the outputs, using Ring Confidential Transactions [Noe15].

To support multiple assets, we will create a process by which designated operators may collaboratively mint the new token types to the MobileCoin blockchain. This process is designed for both security and auditability so that any third party can understand the on-chain supply of the minted tokens. We will also create a corresponding auditable burn process. These together enable the creation of fully-collateralized stablecoins on our chain. Note, the process outlined in this Section 2.2 is a high-level summary and each particular token type may have some variations in the structure required for mint and burns.

The birds-eye view is that after a set of **Governors** (Section 2.2.3) has been agreed upon by the MobileCoin consensus validator node operators, a new **Minting Configuration** (Section 2.2.2) is submitted to specify the signers who are allowed to mint the new asset type on the MobileCoin blockchain.

Once the governors are chosen, collateral needs to be obtained. When collateral has been obtained (via the "purchase" process), the governors acknowledge it and collectively agree to mint a corresponding amount of stablecoin. They submit a **Minting Transaction** (Section 2.2.1)) for the appropriate amount, which produces a new TxOut on the Mobilecoin chain. This can then be used in transactions normally. Later, if the redemption process is triggered, any amount of stablecoin which a user possesses may be **verifiably burned** (Section 2.2.4) on the MobileCoin blockchain. The governors then acknowledge this and collectively agree to release a corresponding amount of reserves.

To achieve the set of functionality required to support backed assets on the MobileCoin blockchain, we will add 2 new transaction types:

1. **Multiparty Minting Transaction**, `MintTx`, producing a TxOut which is indistinguishable from other TxOuts on the chain, along with a new type of block contents that include information specific to the token mint that occurred

2. **Authorized Signer Declaration**, `MintConfigTx`, (also known as "Set Minting Configuration Transaction"), producing a new ledger entry: **Minting Configuration**

Verifiable burning simply uses a normal MobileCoin transaction which sends the burned funds to a new designated "burn address," which renders them visible but unspendable.

2.2.1 Multiparty Minting Transaction

Minting occurs when the minters (previously approved by the governors) decide to increase the supply, minting new tokens to a particular address of their choice.

Each of N signers maintains an Ed25519 [Ed2] keypair:

$$(A_i, a_i)$$

and provides their public key to be published in the MobileCoin blockchain as part of a "MintConfigTx".

The contents of the `MintTx` include the full set of signatures of each of M signers over the transaction's `TxPrefix`.

This is a multiparty transaction when M ¿ 1 in the MintConfig which authorized these minters – to create the transaction, they must each generate identical MintTx's and sign it. These signatures can be aggregated before submission.

Each Consensus Validator node learns the approved signer lists via the MintConfigTx's in the blockchain. When a Consensus Validator node sees a `MintTx` with an M-of-N set of signatures, it considers the transaction valid, and consensus proceeds as normal (see Section 3). When a quorum is reached, and the nodes externalize the transaction, the minted TxOuts are written, along with a new block contents section, `mint_txs`, which contains:

- The Token ID (only some Token Types allow minting, which is verified by the enclave)
- The amount being minted
- The destination's public subaddress view key
- The destination's public subaddress spend key
- the Ed25519 multisig signature set, including the total M signatures required for the $M-of-N$ threshold, specified by the `MintConfigTx` previously accepted and written to the blockchain
- Nonce: random value to protect against replay attacks
- Tombstone block: block index after which this transaction will be discarded by consensus, to prevent the transaction from being nominated indefinitely

When the new TxOut is created based on a MintTx, the 'tx_private_key' which is the source of entropy which blinds the TxOut cryptographically, is derived deterministically based on a hash of the parent block id, the set of mint tx's, and other on-chain information. This is necessary because,

- The 'tx_private_key' for each minted output must be created deterministically in the enclave so that all consensus nodes generate the same block.
- If 'tx_private_key''s are not unique across all TxOut's, it can cause 'tx_out_public_key' to collide, but consensus and the ledger enforce that the 'tx_out_public_key' is unique per TxOut.

However, this source of "entropy" can in this case be fully deduced from data visible on the chain, so it is always possible to confirm that when a MintTx appears in the chain, the corresponding output that it created has the correct value, without just trusting the enclave.

When submitting a `MintTx`, we include a nonce to protect against replay attacks, and a tombstone block to prevent the transaction from being nominated indefinitely, and these are committed to the chain. (For example, in a bridge application, this nonce may be derived from records on the source chain, to ensure that each deposit on the source chain leads to at most one mint.)

The block, and the latest `MintConfigTx`, ensure that the blockchain is therefore a self-contained document that can be audited to confirm that the minting was sound. In addition, minting is immune to replay attacks, meaning the same transaction cannot be submitted multiple times.

Remark 1 (Privacy Properties of Minted Transaction Outputs) *Once the TxOut is minted, it is exactly the same as any other TxOuts on chain, with the caveat that the recipient view and spend public keys were published in the block's* `MintTx`*.*

TxOut's are produced from MintTx deterministically and anyone can verify that the amount actually in the minted TxOut corresponds to the amount and token type of the MintTx that created it.

However, no one can tell from the blockchain when this TxOut is spent, nor exactly what TxOut's it is used to create when it is spent. Using a minted TxOut in a transaction does not pose any privacy threat, except in threat models where the adversary may have access to the user's rings. In applications where those thread models are considered significant, it may be reasonable to ensure that one self-payment transaction happens before performing a payment using the minted output.

Remark 2 (Set of Signers vs Threshold Signature) *An alternate proposal would be to use a Threshold Signature, such as Flexible Round-Optimized Schnorr Threshold Signatures (FROST) [KG20]. The reason we do not intend to use Threshold Signatures at this time is that we consider the explicit list of signers to be a desirable feature for auditability. In a threshold signature, the signer public key is a composite of all the participants, and it is not known exactly which M of the N participants participated to create the signature.*

2.2.2 Declaring Authorized Signers: `MintConfigTx`

To prepare the consensus validator nodes to validate minting transactions, they must be aware of the set of N authorized signers, and this set needs to be published to the blockchain for security, auditability, and recoverability.

The **Minting Configuration**, `MintConfigTx`, contains one or more MintConfigs, each of which provides (limited) authority for a set of minters to mint tokens. In greater detail:

A MintConfigTx specifies:

- TokenID: the token type to which this configuration applies
- Global mint limit: the total amount that can be minted in this configuration
- Nonce: A value which protects against replay attacks
- Tombstone block: block index after which this transaction will be discarded by consensus, to prevent the transaction from being nominated indefinitely

Each MintConfig specifies:

- Signer set: the set of N signers, $\{A_i, A_{i+1}, \ldots, A_N\}$
- Threshold: the number, M, of signers required per `MintTx`
- Mint limit: the total amount that can be minted by this signer set

All of the values above are committed to the blockchain and will be used when verifying `MintTxs`.

Consensus validator nodes perform consensus when a `MintConfigTx` is proposed via an admin endpoint on the nodes. This provides flexibility to the node operators and keeps the network resilient so that a quorum is reached to allow a new minting configuration, while ensuring liveness of the network, so that the network need not restart in order to change the minting configuration.

Consensus validator nodes use the latest minting configuration to validate `MintTxs`, and a node in catchup would use the latest `MintTx` it has seen.

Remark 3 *Note, however, that nodes in catchup mode do not participate in consensus. The network is already resilient to nodes in catchup, and the minting procedures are similarly resilient.*

2.2.3 Governors: The Signer Set for a Consensus Validator

The consensus validator nodes must maintain a set of "master minters," also known as **governors**, who can sign the `MintConfigTx` transactions. The governor set is maintained in the node's startup configuration file, and it is the responsibility of the node operator to perform due diligence on any governor they are adding to the set. The governor set is also per-token, so different tokens can have different governance.

The governor set is a set of Ed25519 public keys, and a `MintConfigTx`'s `TxPrefix` must be signed by the threshold number of governors required by each node in a quorum of nodes to establish a new Minting Configuration for the network.

The consensus nodes ensure that the governor set is valid in two ways:

- The governor set, as it appears in the startup configuration, must be signed by the "minting trust root" which is baked into the enclave. This key is ultimately controlled by the MobileCoin Foundation.
- The consensus nodes ensure that there is a network-wide agreement on the governor sets, because all of this configuration is hashed into the responder-id. Any two nodes with different hashes here will fail to attest to eachother. This means that in a functioning consensus network, all the nodes agree about the governor sets of all the tokens.

2.2.4 Verified Burning

The verifiable burn address is generated in the following way:

- First, the view private key is determined as a specific known constant Ristretto scalar. (This is created by hashing a fixed string, and then interpreting the resulting bytes as a number and reducing it modulo the curve25519 prime order.)
- Second, the spend public key is determined as a specific known constant Ristretto point. (This is created by hashing a fixed string to the Ristretto elliptic curve.)
- Finally, the view public key is determined by multiplying the view private key by the spend public key. (This is how the view public key is derived for all subaddresses in MobileCoin.)

To burn funds, a normal transaction is created which sends an output to the burn address.

To verify the burn, simply use the view private key and view-key match this TxOut as normal. Note that memos as described in MCIP #30 can also be decrypted.

Because the spend public key is created by hashing-to-curve, it is infeasible for anyone to determine the spend private key corresponding to this subaddress – more specifically, it is as hard as discrete log over Ristretto. Therefore, funds sent to the burn address can never be spent.

Verifying the burn serves several purposes:

- If a burn takes place as part of a redemption process, the verification of the burn can serve as the trigger to release funds. (Potentially, the memo field on the burn TxOut can be used to direct the destination of the released funds.)
- By tallying up all mint and verified burn operations, a third party auditor can determine the total supply of any token on our chain.

2.3 Multiple Asset Fees

The startup configuration for the consensus validator node includes the `TokensConfig`, which specifies the following per token:

- Token ID
- Minimum fee
- The list of governors who can change the Minting Config for this token

Every transaction's fee is paid in the *same* token type as the inputs and outputs of the transaction[5].

2.3.1 Transaction prioritization

Because different Txs have fees denominated in different currencies, and these currency types are intended to remain secret, there is additional complexity around how to prioritize transactions now, which was previously done based on the (MOB) fee of each Tx. These priority queues exist outside of the enclave and one of our design goals was not to change this architecture.

To resolve this, we introduce a new concept called "priority" of a Tx.

Definition 2.1 (Priority) *The* priority *of a transaction is equal to*

$$\mathit{fee\_value}/(\mathit{minimum\_fee\_for\_fee\_token\_id}/128)\ .$$

That is, first we look up the Token ID of the fee, and the minimum fee for this Token ID. Then we divide this value by 128. (It is required by the consensus network that this value is divisible by 128 and it is a configuration error if it is not.) Then, we divide the fee value (in smallest representable 'u64' units) by this, and round down to the nearest integer.

[5] This is different from many other blockchains which always charge the fee in the "base" token of the network. As described in the intro, we do this differently in order to improve the user experience.

Intuitively, the priority captures approximately how many multiples of the minimum fee this transaction paid.

The priority of a Tx can be computed in constant time in the consensus enclave. Note that it can also be computed without the possibility of numeric overflow.

When the consensus enclave exposes a reference to a Tx which is sufficient for prioritization, where previously it exposed the fee, it now exposes the priority. (Note that exposing the fee can potentially reveal information about the Token ID since different Token IDs may have very different minimum fee values, and most transactions typically pay the minimum fee.)

When multiple Txs are queued, prioritization can now work simply by sorting the priority values rather than sorting the fees as previously.

Historical priority values can also be logged and averages publicized, so that clients can understand what priorities they need to select for their transaction to go through during times of congestion.

To understand the privacy properties of this arrangement, it is useful to observe that

$$\text{fee\_value} = 128 * \text{priority} * \text{minimum\_fee} \ .$$

Strictly speaking, this is only approximately true to within 1 percent. However, if a client selects a fee value that is not perfectly divisible when priority is computed, then the rounding down will cause them to achieve the same priority number they would have if they paid a slightly lower fee. So it is irrational for such clients to do that – they should always pay the smallest fee value that will give them the desired priority number. For such clients the above factorization will hold perfectly, and the priority and minimum fee values can be seen as independent parameters.

The minimum fee depends on the Token ID, and the priority does not. The priority is the only component that affects transaction ordering, and the Token ID does not.

Therefore, we can set as our goal that, any two Tx with the same priority value (but possibly different Token IDs) are indistinguishable from the point of view of the node operators. We can be confident that revealing the priority value does not undermine this goal and does not reveal anything directly about the Token ID. In this way, prioritization can occur without revealing the token types involved in the transactions in a block.

2.3.2 Fee aggregation

The MobileCoin enclave performs an optimization called Fee aggregation.

A Tx in the MobileCoin blockchain lists a fee value and a fee Token ID in its TxPrefix. However it does not include directly a TxOut for the fee. Instead it is the job of the consensus enclave to add that output when the block is finalized, and possibly, merge that output with any other fee outputs in the block, to avoid adding unnecessary fee outputs to the blockchain. When the blockchain is under load, this may reduce the total size of the blockchain by about one third.

When MOB is the only currency, fee aggregation is quite simple – we just add the fee for each Tx in the block, and produce exactly one TxOut of that value for the fee recipient.

When there are multiple fee currencies, the problem is considerably more complicated because:

- For each Token ID that is actually used, there must be at least one fee output.
- If the number of fee outputs depends on which Token IDs the users select, then this is potentially a source of leakage about their choices.

For example, if two users submit transactions that end up becoming the only two transactions in a particular block, but they can see that there is only one fee output for the block, then they both would know that the other user must have used the same Token ID that they used, undermining confidentiality.

To avoid making such inferences possible we took the following approach:

The number of aggregated fee outputs is always the min of:

- The number of transactions in the block
- The number of Token IDs that can be used as fees

Note that this does not depend on the Token ID choices of the users. The number of different Token IDs actually used by those transactions may be smaller than the above number. In that case, additional zero value fee outputs are minted (in order to protect privacy).

The implementation in the enclave is hardened against side-channel attacks – we aggregate fees per Token ID across the block in constant-time. We select the required outputs to mint (including zero padding) also in constant-time. Then we mint these outputs also in constant-time. This prevents anyone from inferring whether any transactions in the block had related Token IDs (without penetrating SGX).

In terms of on-chain performance, the benefits of the earlier fee aggregation scheme are largely retained:

- In blocks with only one transaction, there is still only one fee output.
- In blocks with many transactions, the number of fee outputs is at most the number of Token IDs. (This is the best possible outcome since it's possible that every Token ID is actually used in the block and so there would have to be a fee output minted for it.)

3 MobileCoin Multi Transaction Type Consensus

To support the new transaction types, the consensus protocol is updated to perform Federated Byzantine Agreement on the new types as well as the old.

MobileCoin's consensus protocol now consenses not only over transaction hashes for transactions that live only in the enclave, but also over MintTx and MintConfigTx objects. (Note that since these are intended to be published transparently to the blockchain, there is no need to hide them in the enclave during validation.) During SCP, each transaction is validated appropriately, and a block containing any number of the three different types of transactions is proposed. During the form block operation, the enclave validates everything a final time.

MintTx and MintConfigTx do not contain fees, so they are not prioritized in the same way as standard Tx's. Valid mint and mint config txs are immediately proposed via SCP rather than being queued. This is not considered a DOS vector at this time because proposed MintTx and mint config txs with invalid signatures are immediately rejected and neither queued nor nominated.

- If a minter authority is abused in a way that harms network performance, the Governors are in a position to sign a new MintConfigTx to remove the offending minters.
- If a Governor authority is abused, the foundation is in a position to sign a new `TokensConfig` to remove the offending Governors.

4 Auditor

The "reserve-auditor" is a program which audits the supply of any of the new tokens.

The reserve-auditor only performs completely transparent calculations, and anyone can run it without access to any secrets.

4.1 MobileCoin ledger

The primary function of the reserve auditor is to continuously audit a local copy of the ledger. It does not sync the ledger on its own, so to use it you must run mobilecoind or full-service.

The reserve auditor tracks the following about Mint Transactions:

- How much was minted of each token type
- Whether the signatures were actually valid
- Whether the signatures correspond to a valid MintConfig that was in-force at that time
- Whether the minting limit was exceeded

The reserve auditor also continuously scans TxOut's using the verifiable burn address view key. Thus it can determine

- how much of each token has been verifiably burned
- at what points in time.

This enables the reserve auditor to calculate the total supply of all minted tokens, at all points in time. (This information can be queried via a grpc API.)

Unexpected events such as invalid mint transactions and negative supply values trigger a warning. These warnings are exposed via Prometheus metrics to allow easy alerting.

4.2 Gnosis Safe

The reserve auditor also supports optional integration with a Gnosis safe, for any stablecoins whose reserves are held in Gnosis safes. (A Gnosis safe is an Ethereum smart contract which supports multisig functionality for deposit and release.)

When using the Gnosis safe integration, the reserve-auditor uses the Gnosis API to track changes to the safe's balance.

It checks that:

- Deposits to the Gnosis safe correspond to subsequent mint operations on the MobileCoin chain.
- Verfied burns on the MobileCoin chain correspond to withdrawals from the Gnosis safe.

For this integration to work properly, mints and burns are done in a specific way:

- MintTx nonces must be created in a specific format that points to the Ethereum tx hash of the deposit to the safe.
- Withdrawals from the Gnosis safe need to be done in a specific way to record a link to the TxOut public key of the burn TxOut that triggered the withdrawal.

This tool makes it easy for anyone to verify that the reserves for a MobileCoin stablecoin based on a Gnosis safe have been successfully and faithfully bridged to the MobileCoin network at all times.

5 eUSD: Collateralized Asset Bridge Operation

The first asset created using aforementioned confidential tokens functionality is eUSD.

To support the fully collateralized backing of eUSD, the minting and burning of assets is tied to the operations on the backing asset's chain. By utilizing multisig, the bridge operator can provide multiple levels of access and approval shared among multiple parties in order to facilitate locking/unlocking and minting/burning on either side of the bridge.

The initial deployment of eUSD will be a manual bridge, tied to an Ethereum Gnosis Safe contract containing a basket of fully collateralized stablecoins.

For technical details on authorized minting, burning, governorship, and declaring authorization of signers, review Section 2.2.

5.1 Wrapping eUSD: from Ethereum to MobileCoin

To wrap eUSD tokens on the MobileCoin blockchain, the bridge operators monitor the Gnosis Safe contract on Ethereum, and when a deposit is made by a verified entity with minting permissions, the operator facilitates a multisig transaction by organizations whose signing keys have been verified by the blockchain's consensus and included in the configuration of a quorum of node operators.

5.2 Unwrapping eUSD: from MobileCoin to Ethereum

To unwrap eUSD tokens and release the funds from being locked in the Gnosis Safe, the bridge operator monitors the verifiable burn address, and facilitates a multisig transaction from the Gnosis safe by signers who are pre-approved by the bridge operator to release funds.

5.3 Bridge Security

One of the benefits of a manual bridge are the higher degrees of security with which the signing keys can be managed in either direction, and the facilitation of activity by pre-approved parties. In addition, replays are prevented by nonces and proper use of ed25519 signing.

5.4 Governance

The governance of these minting keys is shared between the MobileCoin Foundation, who approves a set of governor keys, the bridge operator, who facilitates transactions by approved parties, and the approved parties for minting eUSD to the MobileCoin chain or releasing assets from the Gnosis Safe contract.

When a new configuration is proposed for the `TokensConfig` file, for example, to introduce a new asset, the governors for the new asset provide their public keys and a threshold for approving LPs. The MobileCoin Foundation coordinates the verification of `TokensConfig` with all of the involved parties, and signs the configuration file. The decentralized node operators can then choose whether or not to include the `TokensConfig`. As the `TokensConfig` contents are hashed and included in the responder ID for node-to-node attestation, any node that is running a `TokensConfig` file different from a quorum of nodes on the network will be treated as a byzantine actor.

The governors then approve the Liquidity Providers (LPs) who, along with the bridge operator, authorize wrapping and unwrapping. In order to mint on the MobileCoin chain, or release the backing asset from the Gnosis Safe contract, a threshold including both LPs **and** bridge operators must sign the transaction.

6 Appendix A

6.1 Impact on MobileCoin Fog and Wallets

The impact of these changes on MobileCoin Fog is minimal: the `masked_token_id` field must be piped through Fog Ingest so that when users get their TxOut's from Fog, they can get this field as well and properly reconstruct their TxOut's. This is just a mechanical change.

Most client software is similarly only minimally impacted. The main thing is that clients must not ignore the Token ID fields, and must not mix tokens when performing input selection to build a transaction, or mix tokens when computing the balance. For components like SDKs they can choose how to evolve their APIs to avoid breakage of software that is downstream of them. We expect that most wallet software will develop in two steps, one in which it filters out and ignores TxOut's with nonzero Token IDs and continues to only support MOB as before, and one in which it fully supports checking balance in and transacting in additional Token IDs.

References

[AF] Aram Jivanyan Aaron Feickert. Spats: User-defined confidential assets for the spark transaction protocol. *https://eprint.iacr.org/2022/288*.

[Alg] Algorand. Algorand standard asset. *https://www.algorand.com/Feature*

[Car] *Cardano. Learn about native tokens.* https://docs.cardano.org/native-tokens/learn.

[Cir] *Circle. Usd coin.* https://www.circle.com/en/usdc.

[Com] *Compound. Compound ctokens.* https://compound.finance/docs/ctokens.

[dHLA16] *Henry de Valence, Mike Hamburg, Isis Lovecruft, and Tony Arcieri. The ristretto group.* https://ristretto.group/*, 2016.*

[Ed2] *Ed25519. Ed25519: high-speed high-security signatures.* https://ed25519.cr.yp.to/.

[ERC] *ERC. Erc20.* https://ethereum.org/en/developers/docs/standards/tokens/erc-20/.

[Haw] *Josh Hawkins. Major update to usd coin (usdc).* https://www.circle.com/blog/major-update-to-usd-coin-released-today-advancing-digital-dollar-stablecoin-usability-security.

[KG20] *C. Komlo and I. Goldberg. Frost: Flexible round-optimized schnorr threshold signatures.* IETF*, Crypto Forum Research Group:draft–komlo–frost–00, 2020.*

[koe21] *koe. Mobilecoin governance, fees, and supply.* https://mobilecoin.com/news/mobilecoin-governance-fees-and-supply*, 2021.*

[Mak] *MakerDAO. Makerdao.* https://makerdao.com/en/.

[Noe15] *Shen Noether. Ring confidential transactions.* IACR*, 2015.*

[Pax] *Paxos. Pax dollar.* https://paxos.com/usdp/.

[Res] *Reserve. Reserve.* https://reserve.org/.

[Tru] *TrueUSD. Trueusd.* https://trueusd.com/.

[Vol] *Gian M. Volpicelli. Terra's crypto meltdown was inevitable.* https://www.wired.com/story/terra-luna-collapse/.

[WBT] *WBTC. Wrapped bitcoin.* https://wbtc.network/.