What is Enshroud?

21 April, 2025

What does Enshroud do?

Enshroud is a software technology that provides privacy as a service on smart contract blockchains. Its purpose is to use encryption to obscure (cast a *shroud* over) spend transactions on Layer 1 or Layer 2 blockchains, without the need for a specific privacy-supporting coin or blockchain. Enshroud lets users make private spends of tokens even on public blockchains such as Ethereum, and other EVM-compatible chains.

How does it work?

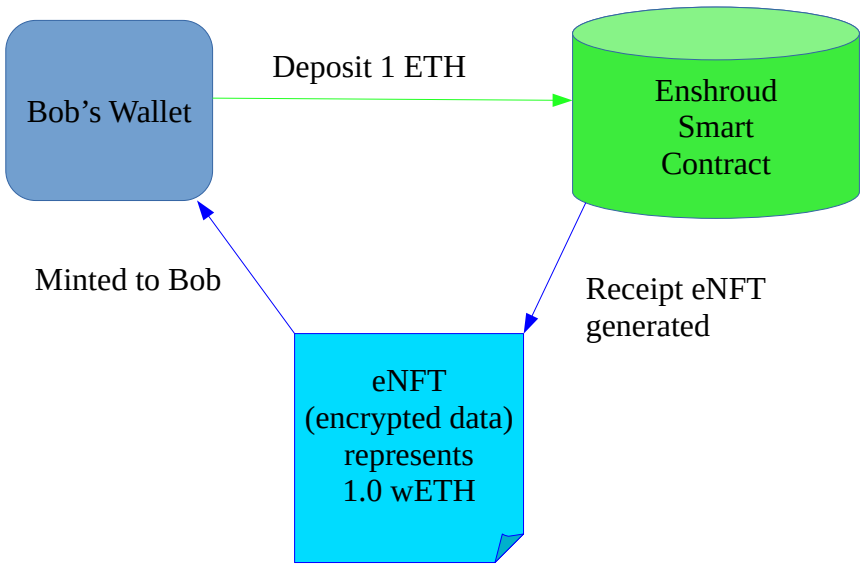
Enshroud uses NFTs (non-fungible tokens) to represent balances of deposited fungible tokens (such as ERC-20s). A typical NFT acts like a transferable title to a piece of artwork hosted at a URL, such as a picture of an ape, or a kitty, a pirate, etc. Enshroud's NFTs act as a title to a piece of encrypted data on the blockchain's event log, which represents a vaulted asset balance. Instead of a JPEG image, what's stored is a small encrypted blob, signed by validators, which only the owner can decipher or use.

So instead of: "Here, I spend to you this nice little picture of an ape smoking a pipe," it becomes: "I spend to you this amount of wrapped bitcoin." (For example.) Since the NFT data is encrypted ("enshrouded"), no one else can see the details of what was really being transferred when the NFT got minted. Enshroud *eNFTs* (encrypted NFTs) conform to the ERC-1155 NFT standard.

A decent analogy is sending encrypted email. Regular token spends in the open are like unencrypted emails. Enshroud spends are like encrypted emails: only the recipient can read it. Ethereum (or other similar account-based blockchains) would be like the email delivery system in this analogy.

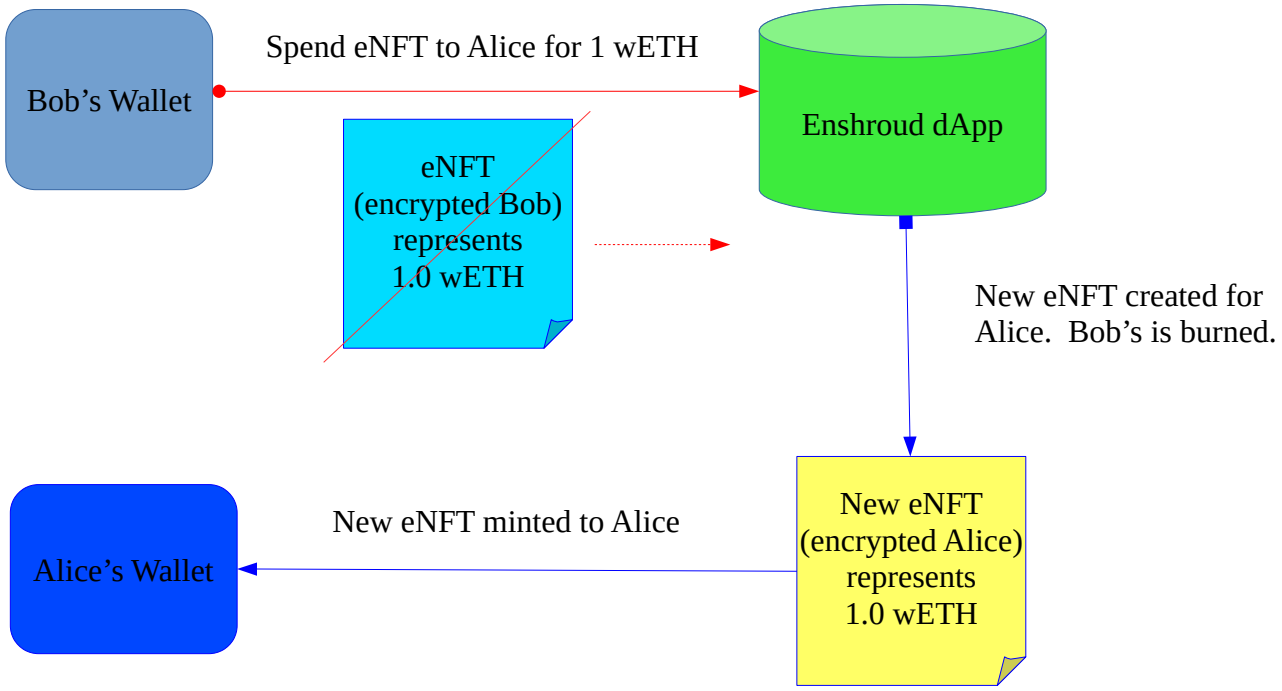
So how is this useful? Examples, please.

Sure. Bob wants to send 1 ETH to Alice privately. First, he deposits 1 ETH into the Enshroud platform. He then mints himself an eNFT using his deposited balance. This eNFT is encrypted to a unique key, so only Bob can read it. Once he opens it, indeed it says it's for 1 wETH.



(Actually the encrypted metadata is generated by Enshroud's Layer 2 servers, and stored in the event log by the smart contract. The smart contract mints the eNFT to Bob's wallet address. We're keeping the diagrams simpler by omitting moving parts. Also, we're ignoring any fees or gas here. Note that wETH = wrapped ETH, an ERC-20 form of ETH to which ETH deposits are automatically converted.)

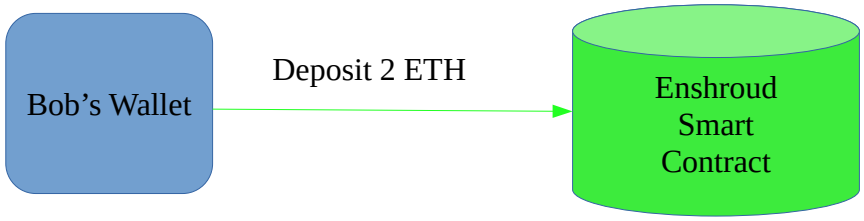
Next, Bob spends to Alice's wallet. He sends Alice 1 wETH via Enshroud, and funds his payment with his eNFT for 1 wETH. The system burns Bob's eNFT and mints a new one, encrypted using a different key for Alice, for 1 wETH and sends it (mints it) to her wallet address.

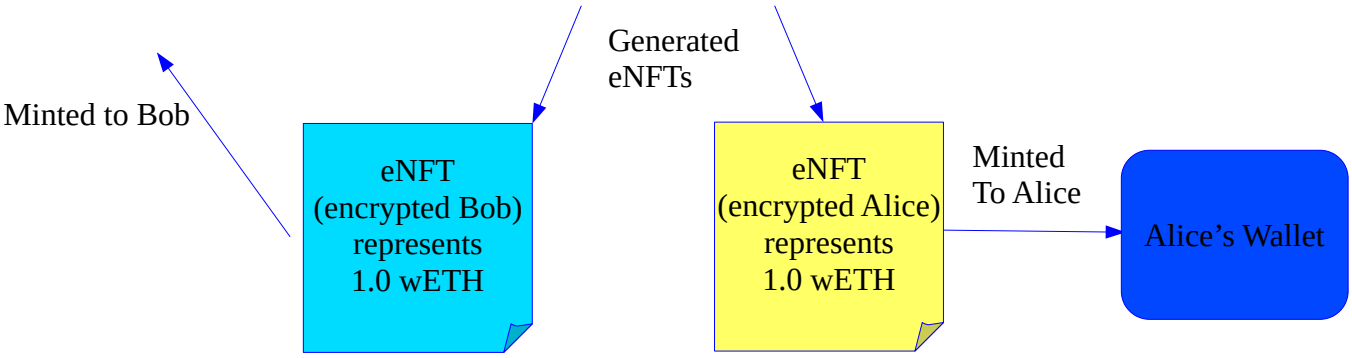


Okay that's nice, but if anyone was really paying attention they could infer that Bob's secret eNFT must have been for 1 wETH, because that's how much his deposited balance credit dropped within the Enshroud smart contract when he minted it. Likewise when he made a spend to Alice, it must have been for the very same amount, because he didn't receive another eNFT back as change.

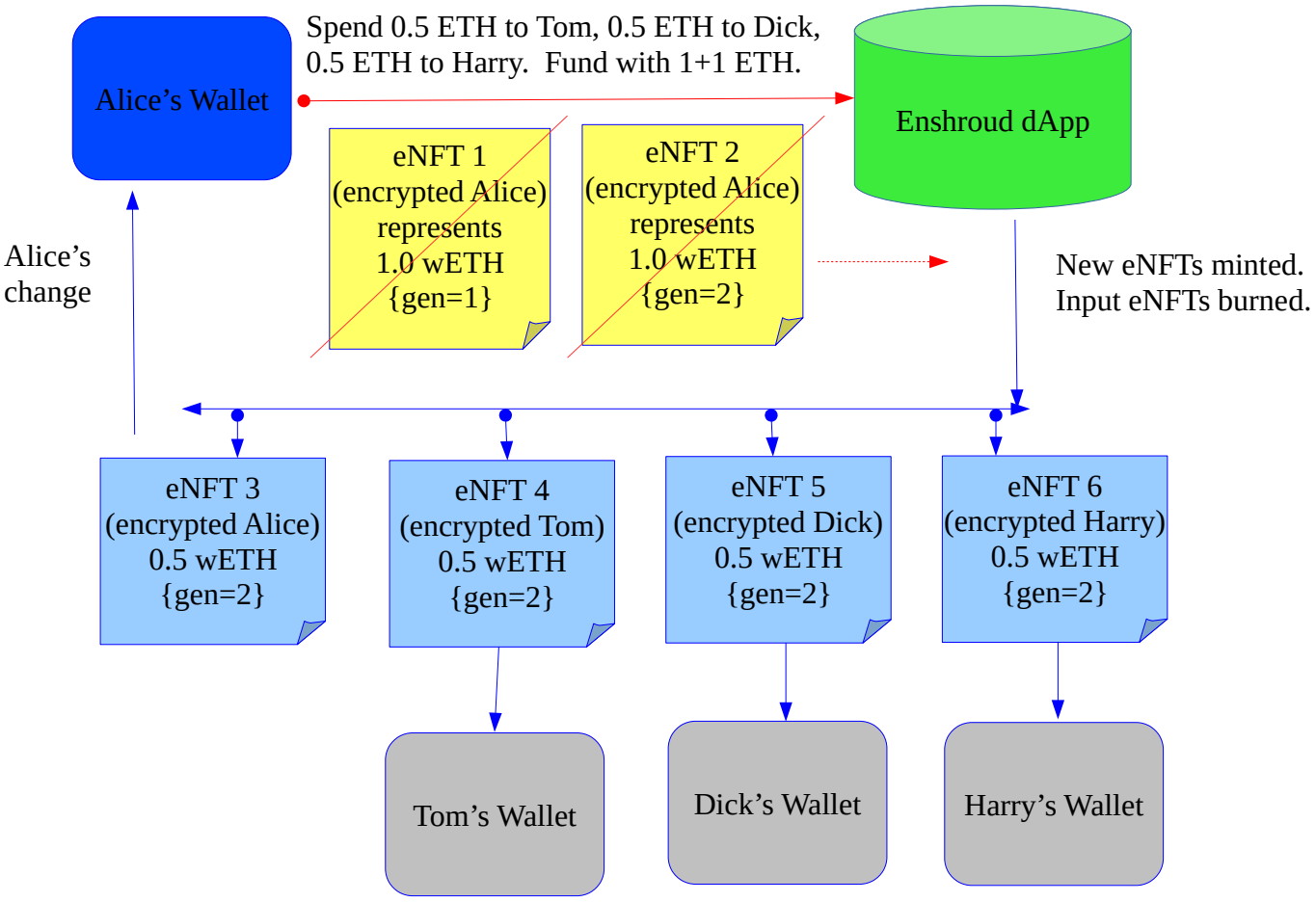
Being a fairly smart guy, Bob realizes that he hasn't accomplished much in the way of privacy by paying Alice this way. So next time he owes Alice 1 wETH, he does it differently.

Bob deposits 2 ETH, and mints to two addresses, his own and Alice's, each to receive 1 wETH. Now the outside observer knows that the two receipt eNFTs must add up to the 2 ETH Bob deposited, but has no way to know what portion Alice received or how much Bob got back.





Now suppose Alice owes 0.5 ETH each to Tom, Dick, and Harry. As we know, she now has two eNFTs totaling 2 wETH. She uses these as inputs to spend 0.5 to each of the three guys, and gets 0.5 change back for herself. So two eNFTs are burned and four new eNFTs are issued. To the observer, these four new eNFTs must sum to at least 1 wETH, but less than 3. Why? Because the amount of Alice's second eNFT was ambiguous: it had to be bigger than 0 but less than 2 (the max Bob could have sent her).



The more times value circulates within Enshroud, the more difficult it becomes to guess at the values of individual eNFTs. In recognition of this, each eNFT is labeled with a secret "generation" value. The generation of an output eNFT is always the lowest generation of the inputs that funded it, +1. Bob's original deposit eNFT had generation=1 (just deposited). The first one Alice got from Bob had

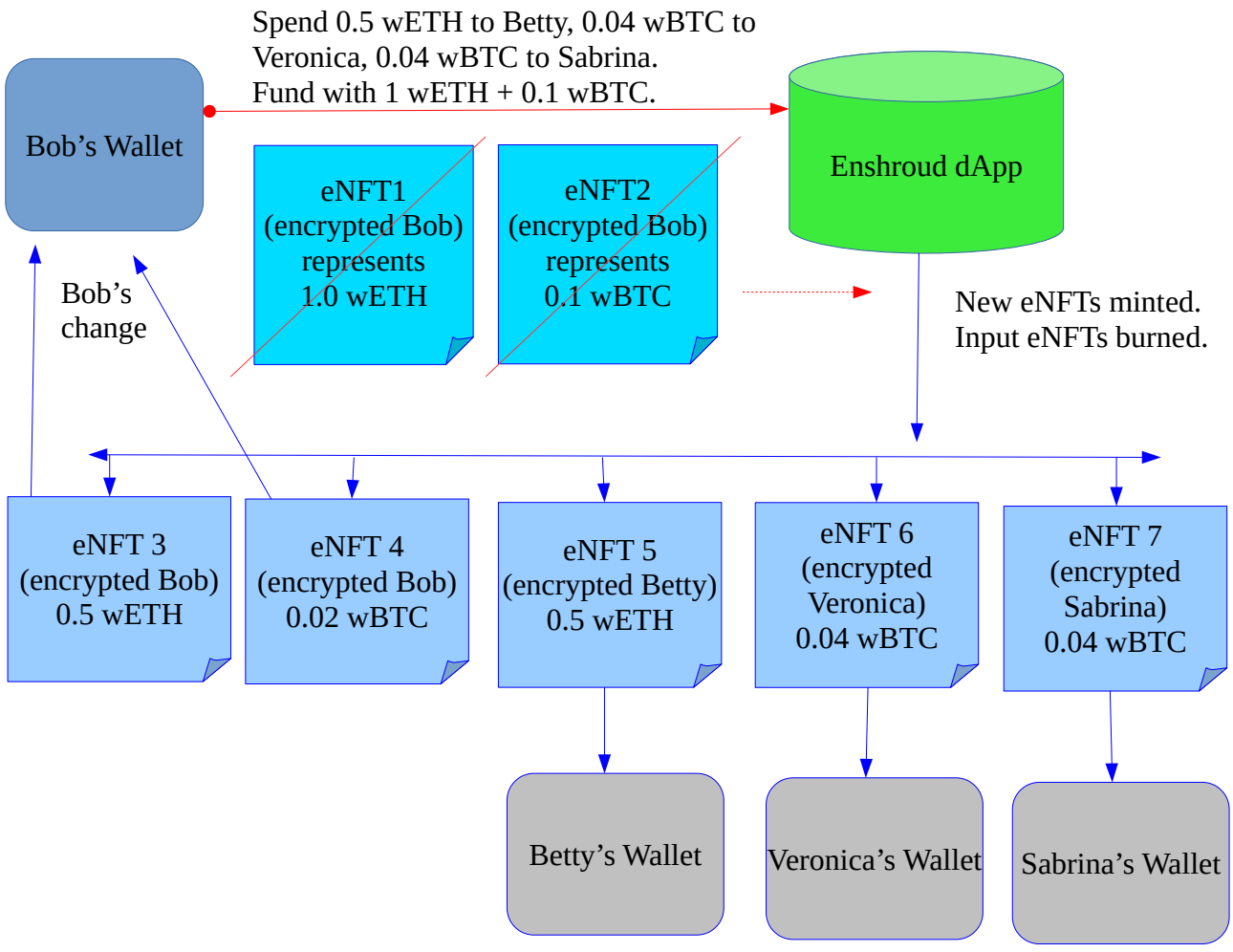
generation=2. Bob's second deposit created two more eNFTs of generation=1, for him and for Alice. So the four eNFTs created in Alice's transaction have generation=2, one more than the lowest of her inputs, which was 1.

Mix it up some more!

Okay. You probably noticed that Enshroud lets you do spends to multiple people while minting from a deposit. That's efficient, because you can do everything in one transaction (which saves gas). But it's sub-optimal for privacy. It's actually better to make your deposits first and mint the eNFTs to yourself, then make your spends. One reason is that asset types aren't revealed in spends, and in fact Enshroud even lets you mix and match your asset types in the same transaction!

Suppose Bob deposits 2 ETH, and tells the system to mint 2 eNFTs for it, each one for 1 wETH. He also deposits 0.1 wBTC (wrapped bitcoin, which is an ERC-20 token representing bitcoin on Ethereum) and mints a separate single eNFT for that, for the whole amount.

Now, Bob wants to spend 0.5 wETH to Betty, 0.04 wBTC to Veronica, and 0.04 wBTC to Sabrina. He uses two of his eNFTs as inputs (one of the ones for 1 wETH, plus the 0.1 wBTC one). A total of five new eNFTs get created: three for the ladies, plus one of each asset type for Bob's change. After this, Bob will have two eNFTs totaling 1.5 wETH (1 + 0.5) and another one for his leftover 0.02 wBTC.

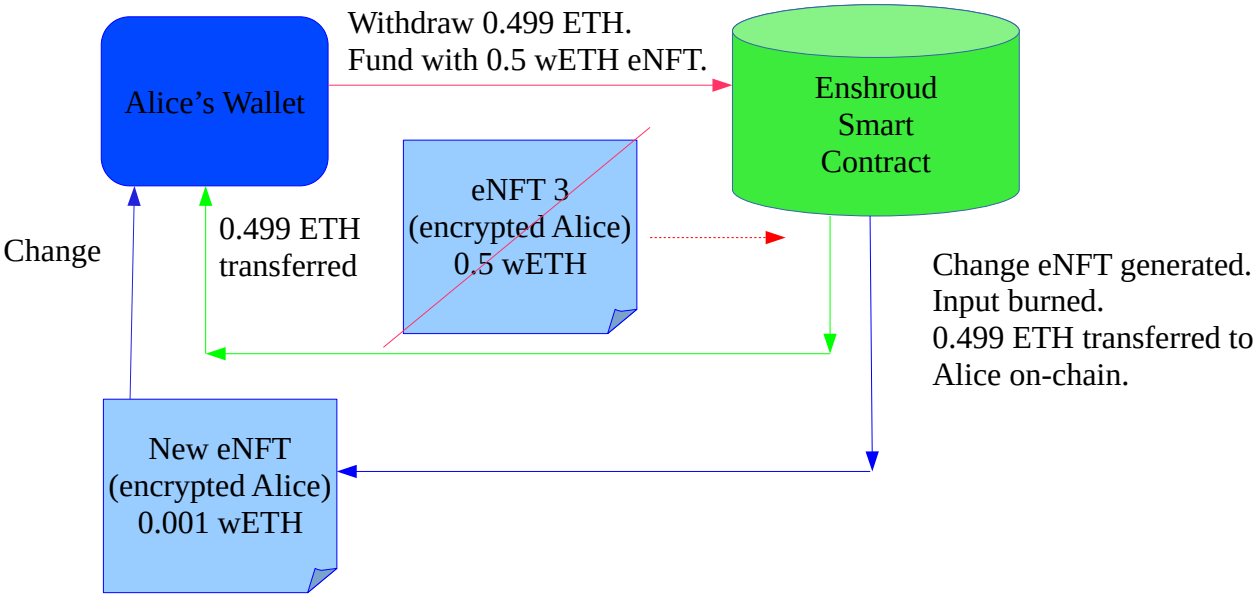


From the outside observer's perspective, this is now almost completely ambiguous, despite the fact that Bob is spending tokens he recently deposited. It isn't clear which asset type Betty, Veronica, and Sabrina received. It could have been either wETH or wBTC in all three cases. The amounts are also obscured, especially because Bob minted two eNFTs for his ETH deposit instead of one. He could even have spent only wETH, or only wBTC, to the three ladies, and spent the other asset 100% back to himself. (Yes, it's legit to include your own wallet in a spend.) There's just no way to tell for sure.

If you know how Bitcoin works, you may have noticed by now that eNFTs work a lot like UXTOs, except that only the owner can see how much value is there. Observers can only estimate from context.

What happens on a withdrawal?

Excellent question! On a withdrawal, a user redeems (burns) one or more eNFTs (all of the same asset type) to fund their withdrawal to their wallet, in the backing token type. For example, suppose Alice takes her 0.5 wETH eNFT that she received as change from her spend to Tom, Dick, and Harry, and burns it. That eNFT is destroyed, and Enshroud sends Alice 0.5 actual ETH (visible on the blockchain, auto-converted back to ETH). This act implicitly reveals how much Alice's eNFT was worth. By inference, it clarifies (a bit) the total which was paid to Tom, Dick, and Harry, because Alice's change can now be subtracted from it. But it still doesn't reveal how much each one of them received. Alice could have improved her privacy by withdrawing 0.499 ETH and getting a change eNFT for 0.001 wETH. (Or included as inputs more eNFTs she obtained from other transactions, etc.) We'll diagram Alice being more clever:



Bottom Line: Spends are > Private than Mints or Burns

In summary, mints and burns are done in single asset types, and the total of all eNFTs minted or burned during those transactions can be inferred from the amount which was transferred in the clear on-chain.

In contrast, spends of assets within Enshroud can be mixed and matched in all sorts of ways, and the longer they circulate through successive transactions (the higher the "generation" of the eNFTs), the greater the degree of uncertainty. Both mints and spends support a "memo" line message to the payee.

Note on Keys

All eNFTs are encrypted using AES-256 symmetric keys generated only for use with that specific eNFT. Access to the keys is restricted to the account owner, which is what is meant by "encrypted to Bob," "encrypted to Alice," etc. in the examples above. Keys for burned eNFTs are subsequently purged. Thus only someone who had a record of the keys (as from an old downloaded wallet save made while those eNFTs were current) could carry out a forensic analysis. Since AES keys can be retrieved only by the owning wallet address, that essentially makes it impossible for third parties to track transaction history after the fact without the cooperation of the wallet owner.

Transaction History

So how is Enshroud transaction history made available to users? This is done by automatic generation of encrypted receipts, for both payers and payees, by Enshroud's Layer 2 servers. Because these servers sign the receipt details, they can be used as proof-of-payment. As with wallet eNFT keys, receipts can only be downloaded by their owners. The difference is that decryption keys for receipts persist until the owner explicitly submits a signed request for receipt deletion. Receipts themselves are not stored on the blockchain, and may be stored on decentralized media such as IPFS. While deletion of all copies of the stored encrypted receipt files cannot therefore be guaranteed, destruction of their AES keys guarantees that any surviving copies become unreadable. This effectively means that anyone seeking transaction details (beyond metadata from ERC-1155 *TransferSingle* events) *must* obtain it from one of the parties to a given transaction.

This is as it should be. Warrantless surveillance of financial transactions via third parties without the knowledge of the participants in those transactions is *never legitimate*, and is technically illegal in many (most) jurisdictions. These legal prohibitions are just routinely ignored by government actors.

Taking it to the Next Level

Suppose a liquidity pool existed which took in assets represented by Enshroud eNFTs. So instead of being say ETH/DAI, it would be eETH/eDAI (where eETH = eNFTs representing Ether, and eDAI = eNFTs representing DAI). The balances in this pool would be the sum of the eNFTs in each category. Assume further that the individual eNFTs in each balance are not disclosed, only the totals.

Suppose the pool is an AMM, and a liquidity taker spends an eNFT into it. The AMM gives them back another eNFT of the opposite type. Unless that user was the only one who used it during a given block, it would be impossible to attribute any shift in the published balances solely to that particular user's action. This implies that it isn't possible to deduce which asset the user provided, let alone how much. An observer could know which wallets interacted with the AMM in that block, but not what they did.

Thinking about where this would lead, it quickly becomes clear that Enshroud also has the potential to help privatize DeFi, even on non-private EVM-compatible chains. Food for further thought.