Existing privacy solutions leave traces that make them a clear target for bans and censorship, which as a result hurts fungibility, e.g. blacklisting coins that were used in the banned protocols. Now, imagine singing a private transaction that reveals nothing about your identity and, at the same time, is indistinguishable from any other regular transactions on-chain. Such privacy overlay would be radically more censorship-resistant and uniquely sustainable. Furthermore, anyone analyzing the chain would have to deal with the possibility that any transaction can secretly be a covert privacy-enhanced one. So overall privacy is improved for all users even without coordinated changes in behavior.

In this post, I introduce a way to enjoy covert privacy-enhanced transactions on Ethereum TODAY

. Actually, this solution would work on ANY

public blockchain that supports ECDSA or Schnorr regardless of their smart contract capabilities. For more details please have a look at the full paper and the github repository.

This idea of covert private transactions is actually nothing new: the protocol described here is based on CoinSwap — the idea that was proposed by Greg Maxwell and revisited recently by Chris Belcher. In hindsight, CoinSwap is a non-custodial way of swapping one coin for another one without linking inputs and real outputs on-chain. To learn more about CoinSwap see Design for a CoinSwap Implementation for Massively Improving Bitcoin Privacy and Fungibility

The original CoinSwap idea uses a 2-of-2 multisig to establish a shared intermediary address where coins are deposited by Alice and then withdrawn to Bob's new address. Using smart contracts isn't an option because it's inherently easier to distinguish transactions that involve contract calls. All interactions between parties must happen scriptlessly and offchain.

Similar idea underpins my previous post aboutOffchain and Scriptless Mixer, where we used MPC to mix coins in a shared EOA. Its security, however, was too expensive and we decided not to pursue it further.

This time, 2-party computation (2PC) is used to jointly produce valid ECDSA signatures. A regular 2P-ECDSA scheme like [Lindell'17

](https://eprint.iacr.org/2017/552) isn't enough though, because it can't guarantee output delivery, so nothing prevents Bob from going offline after receiving a valid signature for his tx. Instead, an extended 2P-AdaptorECDSA

scheme from Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability is used to jointly produce adaptor signature, which acts as an atomic lock to release Alice's transaction only when Bob's one is published. To learn more about adaptor signatures see Lloyd Fournier's survey paper on the subject.

CoinSwap can be seen as single-chain atomic swaps with unexpected privacy benefits and similar to other such protocols it must handle refund paths for cases when some party goes offline before finalization. A standard solution here is a hash-time lock contract (HTLC), but again it isn't an option for us. Likely, we can simulate HTLC using verifiable timed commitments

(VTC), which is another cryptographic scheme that hides witness for a certain time and involves zero-knowledge proofs for witness verifiability. In practice, there are two ways to implement VTC: 1) using homomorphic time-lock puzzles (HTLP) and 2) distributed time-lock systems like Drand which I recently made verifiable in myzk-timelock project.

Existing contract-based privacy solutions, such as TornadoCash, also come with the ability to delay withdrawals for an arbitrary time, which makes it harder to perform time correlation, and users' privacy is further improved. VTC allows us to support such functionality here as well: Alice can ask Bob to delay his withdrawal from a shared account for some arbitrary time and to enforce this she will time-lock an intermediary value needed for Bob to complete his withdrawal. Please see section 4.1 of the full paper for more details.

On Ethereum, the primary benefit of having privacy-enhanced transactions is to hide who and when enters and leaves certain dApps, i.e privacy-enhanced on/off-ramping.

However, while originally shared CoinSwap addresses are merely seen as an intermediary pit stop where coins are locked for parties to agree on their further destination, in SπPETs their role extended to support arbitrary contract invocation.

Once funded with ETH by Bob, parties can use 2P-ECDSA

to sign any type of transaction requested by Alice, including contract calls. Again, on-chain there is no direct link that connects Alice to that transaction, all that is seen is Bob interacting with some dApp from his "new" address. Here are some applications where such a distraction would make sense: 1) exchange coins on DEX, 2) purchase or mint NFT, 3) deposit coins into a pool in exchange for liquidity-provider (LP) tokens, etc. This also underpins another relevant functionality that SπPETs is able to support — privacy-enhanced ERC20 operations.

The simplest form of CoinSwap transaction would cost double what a regular transfer does ($2 * 21000 = 42000$

gas). An interaction that includes 3 input multi-transactions and 3 hops routing is expected to provide an excellent privacy guarantee for a cost of $21000 * 18 = 378000$

gas, which is about the same cost as withdrawing from the Tornado Cash pool. However, such complex multi-transaction routed coin-swaps are only for the highest threat models where Bob is assumed to act adversarially. In practice, most users

would probably choose to use just one or two hops.

Finally, I want to emphasize that SπPETs privacy still depends heavily on the overall anonymity set size, so the more Ethereum users leverage it, the better privacy every one of us gets.

That would be it for the post. Head on totimoth-y/spy-pets and try it for yourself. Don't forget to check out thefull paper and let me know what you think.