ASUN*

address sharing using Nostr

A simple uncool solution for the hassles of private address sharing and xpub sharing

* pronounced: “Asoon”. Literally meaning simple, too simple to be called a protocol.

Motivation

Address-reuse has been a huge privacy issue for a long time, and has been frowned apon as such.

On the other hand, Interactive address sharing has been done either manual, which is too much of a hassle for recurring payments and donations, or with payment servers which are not light client (e.g. mobile) friendly, require resources to bring up and maintain and require technical expertise higher than what is expected of average joe to learn.

Of the non-interactive methods, Private Payment and Silent Payment, both have certain problems that make them unfeasible today. PayNym on the other hand “spams” the blockchain, have privacy concerns associated with the notif tx and is extremely expensive to use for single payments (e.g. donations).

How is it done?

- Alice derives her nostr key pair from the path m/696’ using her seed and shares the pubkey P_a with others along with a relay hint.
- Bob(P_b) encrypts the “AddrReq”(or “xpubReq”/”DescReq”) message w Alice’s pubkey (a simple nip4 DM) and sends it to the relay, along with his own “Read” relay hint (which is strongly suggested to be the same as Alices’).
- Alice writes the message: “AddrRes to P_b” along with a preferred-type address. She encrypts the response along with the derivation path with her own pubkey and sends it to the relays (as many relays as she wants).
- After that she strips off the derivation path, encrypts the response using Bob’s pubkey, and sends it back to Bob’s read relay.
- When Charlie(P_c) asks Alice for Address, Alice queries the relays for her own messages addressed to herself. If she finds that she already sent an address to Charlie ($P_c = P_b$) and there isn’t any transaction associated with that address, she resends the same address. If she doesn’t find any messages addressed to Charlie, she sends Charlie the next sibling of the last message’s corresponding address. (note that all messages include timestamps). If Charlie asked for an xpub and Alice had already sent an xpub to him, she will resend it.

Shoulds and Musts

- All users must connect to relays using TOR only.
- Recovery messages (Alice to herself) should be saved on reliable relays, as many as necessary. They could also be saved on cloud service providers.
- For better privacy against traffic analysis, content of the messages should be concated with a random-size randomly generated string.
- Responding to “(x/y/z)pubReq” must be limited to the pubkeys that already exist in Alice’s contact book. All xpubs must be in serialized form and include version bytes.

  Note that the metadata is not encrypted, meaning it’s possible to see who’s sending messages to who, to prevent the transactors to be identified:
- Messages between users is better to be discarded, a relay with a deletion policy is preferred.
- client and relays should support nip45 (client auth). This also prevents spam requests.
- For maximum privacy, users should run their own relays.
- Using the key pair as general purpose Nostr account is preffered. Since the addr exchange messages would be lost in the pool of unrelated messages, It wouldn’t be possible to guess the number of Transactors.

Gap limit problem

“To prevent performance issues, both light and full node reliant wallets limit the amount of addresses without balance that they track on the blockchain.” -BTCPay Server website.

- To mitigate this, ASUN doesn’t let non-contact Pubkeys get more than 1 unused address. XpubReqs won’t be answered by default unless the Pubkey is in the contact list.
- Using ClientAuth(nip42)-enabled relays and demanding a proof of work threshhold (nip13) for each incoming message prevents spam attacks.
- Alice could also set a password that both prevents spam and “the mystery shopper attack”. Each AddrReq must be concated with password to be responded to. Meaning Bob’s normal “AddrReq” would not be answered, Bob has to send “AddrReq{password}”. This also gives alice a way to find out where do her one time payments come from. (multiple passwords could be used simultaneously)

- The last used index will always be in the Recovery messages, Wallets aware of this scheme will always know which addresses they have to look for even with gaplimit=0. For wallets unaware of this scheme, Alice can always check the last index of the last shared address and use that as the gap limit for the wallet.

Derivation Paths

- Using a shared concise system allows us to bruteforce the derivation paths in case of losing the recovery messages.
- For address sharing, We use the first unused hardened “account” (Default=1: e.g. m/84'/0'/1'/0/0) and increment the index for each new address.
- For xpub-sharing, we use the xpub derived from the same path (and incremented index). Resulting in added levels to the derivation paths.
- The reason we don’t want manual address sharing and Asun address sharing to collide is that Eve could ask for an address without ever sending a transaction. The wallet softtware that doesn’t support Asun may give this address to alice as a recieve address. This gives Eve a way to track Alice’s transaction, associating the transaction with the nostr key.

Backward compatibility

No existing wallet uses Nostr, but since the messages are human readable, senders can manually send Requests to the reciever’s nostr pubkey.

Address Requests will be responded by sending addresses dervied from standard derivation paths. Meaning the reciever wallet can see the incoming coins whether itself supports Asun or not(assuming the wallet keeps checking for active accounts). This ofcourse does not include coins sent to an address generated from a shared xpub.

Wallets that support custom derivation paths could be used to spend from the addresses generated by the shared xpub. But a wallet aware of this scheme will have a better UX. Wallet that do not support custom derivation paths but support using xprvs can also be used.

Watch-only wallet

The whole scheme could be boiled down to a watch-only wallet. The only thing needed is the nostr private key (m/696’) and the xpub belonging to the first unused hardened account (e.g. m/84’/0’/1’). And optionally the xpub belonging to the account(s) being currently used outside this scheme.

The watch only wallet acts as a very simplified nostr client. Nostr private key signs the nostr events and encrypts messages. Xpub is used for address/xpub generation.

Losing the nostr private key is equivelant to losing xpub (in an ordinary watch-only wallet), revealing the encrypted messages and transaction history. But with the added risk of impersonation which in most cases could be mitigated (since both the attacker and the victim will have access to the key, the victim could sign messages informing others of the leak).

Backup / Revovery

The Mnemonic (+ Passphrase) words alone are enough for recovery.

Alice derives the nostr key pair(m/696’) and asks the relays for her messages addressed to herself, filters the “AddrRes” and “(x/y/z)pubRes” messages. The derivation path is included with each message.

If somehow alice loses the recovery messages, she could bruteforce her money out by starting at reference points checking the addresses and xpubs and incrementing.

Forward Compatiblity

It’s possible to incorporate bip47 and other proposals into this scheme, but it is not needed in most situations and it damages their wallet’s interoperability. it’s possible regardless.

Advantages over payment server

- The implementation does not have to be online 24/7. Alice answers whenever she’s online, Bob gets the answer whenever he is online.
- It’s not resource intensive, it can be run on mobile devices (which usually have a much more uptime than PCs).
- It doesn’t need technical expertise and the maintaining cost of running a payment server.
- It doesn’t have a central point of failure. The data isn’t stored in one place.

Cons:

- There are privacy issues regarding the metadata that got addressed in previous slides.

Comparison

	notif Tx	UTXO bloat	LightClient /Mobile	Interaction	Recurring Payments	One time payment	Identity layer	Watch Only
PayNym	Yes	Yes	Yes	No	Yes	No	Yes#	No
Silent Payment	No	No	No	No	No	Yes	No	No
Private Payment	Yes	Yes	No	No	No	No	No	No
BTCPay Server	No	No	No	Yes	Yes	Yes	No	Yes
Asun	No	No	Yes	Not at the same time*	Yes	Yes	Yes	Yes

*parties don’t need to be online at the same time and in the case of xpub sharing only the first interaction is needed
#Unlike Asun using PayNym as Identity layer doesn’t benefit privacy, since the number of transactors is always visible on chain it likely damages anonimity.