Fixing Privacy Problems in the Zcash Light Wallet Protocol

Taylor Hornby

Action Items (preview)

- In priority order from top to bottom:
  - **#2 Research spike to investigate feasibility of integrated-in-app Tor on mobile.**
  - **#8 Prototype a PIR implementation with user-configurable privacy setting.**
  - **#3 Constant-bandwidth version of the current wallet protocol**.
  - **#11 PoC traffic-analysis side-channels against lighwalletd and Monero.**
    - This will help us confirm the fixes while generating novel research.
  - #9 Find out what users' privacy needs really are.
  - #4 Research approaches to hide outbound transactions broadcast by lightwalletd.
  - #5 Research spike to decide on note-merging / auto-shielding / etc. defenses to dust attacks.
  - #6 Continue long-term mixnet research for scalable-Zcash, aiming for mobile device compatibility.
  - #1 Validate block headers, proof of work, and the note commitment tree root.
  - #7 Research changes to liberated payments to help improve privacy for light wallets.
  - #10 Implement the download-all-memos defense.

Problem #1: The wallets don’t check headers & PoW

- Lightwalletd (malicious or compromised) can outright lie about transactions.
- You can’t directly steal funds, but double-spending can be just as bad.
    - Example: A malicious lightwalletd operator knows that a product seller is using their lightwalletd to receive payments prior to shipment. The operator sends a fake transaction to their wallet to purchase the product, and the seller doesn’t realize until after they’ve shipped.
- The fix is obvious and simple: validate the headers, check the proof of work, and use darksidewalletd to test the wallets’ responses to active attacks from lightwalletd.

My focus is on privacy problems.

In what follows, I assume this isn’t fixed.

Action Item #1:

Validate block headers, proof of work, and the note commitment tree root.

Problem #2: There is no network anonymity.

- To the extent that IP addresses correspond to real identities, lightwalletd learns who each request is coming from. Attackers eavesdropping on lightwalletd's internet connection learn who everyone is, too.
- *This makes all the other problems worse, because users are identifiable rather than being anonymous or pseudonymous*.
- Fix idea #1: Tor
  - Apparently, Tor is hard to do on mobile. On android, the typical way to do it is to use a separate app which acts as a proxy.
  - We need to do a research spike with mobile devs to investigate if Tor is feasible. I don't understand why it isn't possible to link against a Tor library and bundle Tor in-app.
- Fix idea #2: CDN that doesn't log.
  - If we can't use Tor, what the Brave browser does is configure a CDN that doesn't log.
  - Not an ideal solution, and we aren't in control, since it's a lightwalletd deployment detail.

Action Item #2:

Research spike to investigate feasibility of integrated-in-app Tor on mobile.

Problem #3: Traffic analysis side-channels

- Passive monitoring of lightwalletd’s Internet connection reveals the transaction graph (who’s-paying-who) between all users of that service.
  - Example: lightwalletd is hosted on AWS, NSA obtains a secret warrant to obtain AWS traffic metadata. NSA learns *which IP addresses* are sending *which transactions* to *which other IP addresses*.
- Passive monitoring of wallets’ internet connection reveals when they are sending transactions to each other.
  - Example: ISP can still tell when two users on their network are sending transactions to each other (even Tor’s traffic padding probably wouldn’t against this!).
  - Example: A repressive regime confirms Alice is the leader of an activist group by sending funds to their public donation address and observing the corresponding memo downloads from Alice’s home connection.
- The fix is to make the protocol constant-bandwidth: https://github.com/zcash/lightwalletd/issues/307

Action Item #3:

Implement a constant-bandwidth version of the current wallet protocol.

Problem #4: lightwalletd eavesdroppers see outbound transactions

- After a user submits a transaction, lightwalletd sends it to the p2p network in plain text. A passive observer sees when any user submits a transaction.
  - Example: A small group of activists host their own lightwalletd. The hosting provider learns exactly which transactions were sent by members of that activist group.
  - Example: Without Tor, an attacker eavesdropping on lightwalletd's connection learns who broadcast which transactions.
- This is a problem zcashd shares, and the fixes will be similar.
  - Dandelion++: https://arxiv.org/abs/1805.11060
  - Tor transaction relay network: https://github.com/zcash/zips/pull/391
  - & more. I didn't have time to evaluate these fixes or look for others.
  - Stolon: https://github.com/ZcashFoundation/zebra/blame/stolon/book/src/dev/rfcs/0006-stolon.md

Action Item #4:

Research approaches to hide outbound transactions broadcast by lightwalletd.

Problem #5: Dust attacks reveal when the wallet spends funds

- After sending dust to a wallet’s zaddr, it will spend it by generating a transactions with many outputs. This (roughly) reveals when the funds were spent.
  - Example: Attacker sends dust to a user’s public donation address.
    - On-chain, they find a handful of points in time when it might have been spent.
    - Observing transactions outbound from lightwalletd(s), narrows down which service provider the address owner is using.
- Zcashd shares the same problem, the fix will be similar.
  - https://github.com/zcash/zcash/issues/4332

Action Item #5:

Research spike to decide on note-merging / auto-shielding / etc. defenses to dust attacks.

Problem #6: lightwalletd learns which transactions belong to the wallet

- As soon as the wallet notices a transaction belonging to it, it fetches the memo from lightwalletd.
  - Example: Alice's wallet submits a transaction, then Bob's wallet downloads the memo. Even after fixing all previous problems, lightwalletd still learns the entire transaction graph.
  - Tor helps, but only somewhat, since users can still be identified by:
    - Time they're active, **transparent transactions they send**, or by watermarks that were actively saved to the user's wallet by lightwalletd (e.g. keeping the target user on an even-numbered block).
- I did my best to survey the state-of-the-art research for fixes to this problem. The next slides go over the options.

Fixing #6

Lightwalletd learns which transactions belong to the wallet

We have several options

1. Out-of-band transaction delivery
    a. Mixnet
    b. Liberated payments
2. Ad-hoc approaches
3. Oblivious RAM (in SGX)
4. PIR
    a. Computational
    b. Information-theoretic
5. Everything else I looked at (including just downloading all the outputs)

1. Out-of-band transaction delivery (1/2)

- Mixnet, or other anonymous communication network tech.
  - Deploying a mixnet is a massive project. I recommend lumping it in with Scalability 2021 work.
  - Scalable-Zcash will *require* a mixnet or similar tech, so let’s not duplicate work.
    - Why? Because by encoding messages into memos or ZEC values, scalable-Zcash *is* an anonymous communication network.
    - Known privacy-vs-performance tradeoff limitations proven for anonymous communication networks will also apply to scalable-Zcash.
      - https://arxiv.org/pdf/1812.05638.pdf is state-of-the-art.
      - https://eprint.iacr.org/2013/531.pdf
    - I read https://www.usenix.org/system/files/nsdi20spring_kwon_prepub.pdf. It’s insufficient for our needs, but has good references to prior and orthogonal research.

1. Out-of-band transaction delivery (2/2)

- Liberated payments.
  - We can take advantage of the communication channels users already have set up to send transactions more privately. Ideally, receiving a liberated-payments transaction would be indistinguishable from receiving a normal message through the channel then opening the wallet app.
  - As written, the draft-ZIP doesn't satisfy this goal, and changing it to do so would contradict some of its stated requirements.
  - We could change liberated payments' requirements and design to make them more private for light wallets using the current protocol, but that wouldn't help with the common use case of sending funds to a zaddr, so it should be low priority.

Action Item #6:

Continue long-term mixnet research for scalable-Zcash, aiming for mobile device compatibility.

Action Item #7:

Research changes to liberated payments to help improve privacy for light wallets.

2. Ad-hoc approaches

- Downloading some, but not all of the outputs
  - ZecWallet-lite downloads more outputs than it needs to in an attempt to hide which ones it's interested in, e.g. downloading all outputs in a block.
  - It's not possible to save bandwidth this way and still satisfy any formal definition of privacy. Here's the proof you can't have sender-receiver pair unlinkability:

The setup is two senders S[0], S[1], two receivers R[0], R[1], and for challenge bit b, S[b⊕1] sends to R[0] and S[b] sends to R[1], repeatedly. The senders send transactions at random times. The receivers are active at distinct times, known to the adversary. Either R[0] doesn't save any bandwidth by following the protocol, or eventually R[0] (identified by the time they're active) fails to download a transaction that was sent by S[s] for some s, which confirms that b=s.

A real-world example would be if one of the receivers is an anti-government activist group, the other receiver is a bakery. The government finds out which of the two senders is funding the activist group so they can prosecute them and leave the sender that's buying bread alone.

- This approach is security theatre and we shouldn't use it.

3. Oblivious RAM in SGX -- SGX is not secure.

- Oblivious RAM (ORAM) lets a client access an index into memory without revealing which index.
- On its own, it can’t be used to solve this problem, since the client has to *write* into memory to initialize it. It’s not PIR.
- There’s a proposal to put the ORAM client inside an SGX enclave:
  - ZeroTrace: https://eprint.iacr.org/2017/549.pdf
- The idea was that using ORAM inside SGX defends against SGX side-channel attacks. Unfortunately, SGX’s integrity has been broken as well by using side-channels to extract its signing keys.
- We should not back ourselves into a corner that depends on SGX being secure. At best, this would be a defense-in-depth measure.

4.a. Computational PIR (CPIR)

- PIR lets a client retrieve a record without saying which one.
- Computational PIR is based on computational-hardness assumptions.
- State of the art is XPIR and SealPIR (which is built on XPIR)
  - https://eprint.iacr.org/2014/1025.pdf
  - https://eprint.iacr.org/2017/1142.pdf
- It’s based on the Ring-LWE hardness assumption.
- Ballpark numbers (from the SealPIR paper):
  - 64KB queries, 256KB responses (could be low by an order of magnitude).
  - 0.1s to 2.0s server CPU time per request (database in RAM, uses precomputation).
  - Client CPU is pretty low.
- If this were all we needed to do, it would be great, but **there are two reasons it’s not so simple, which we’ll come back to**.

4.b. Information-Theoretic PIR (ITPIR)

- PIR based on having multiple databases that don’t collude with each other. As long as some fraction of them are honest, none of them learn which record is being accessed.
- State of the art:
    - https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final90.pdf
    - Simpler protocols (described in the same paper) are sufficient if we don’t care about byzantine robustness (e.g. if the wallet uses the commitment tree to check integrity) or continuing to operate when some databases to go offline.
- Perhaps Zcash Foundation runs one database, ECC runs another.
- Similar server and client CPU as CPIR in the usual case.
- Query and response is ~250KB, for two databases of 1M note ciphertexts (currently ~400k). An order of magnitude more to have Byzantine robustness.

4.a,b. PIR Fundamental Challenges

- There are two major problems with deploying PIR, regardless of type.
- Problem #1: PIR queries are big and we just said that we need the protocol to be constant-bandwidth.
- Problem #2: If the wallet makes a PIR query immediately after learning it received a transaction, it informs lightwalletd (and any passive network eavesdroppers) that a recent sapling output belongs to the wallet! The anonymity set is still extremely small!
- To fix these, *timing of PIR queries needs to be divorced from transaction receipt.* **Either the wallet uses lots of bandwidth making frequent PIR queries when there aren't any transactions, or the wallet makes them less frequently, increasing the latency to retrieve memos.**

4.a,b. PIR Fundamental Challenges (2)

- *Timing of PIR queries needs to be divorced from transaction receipt.* Either the wallet uses lots of bandwidth making frequent PIR queries when there aren't any transactions, or the wallet makes them less frequently, increasing the latency to retrieve memos.
- Fixing this properly requires some bandwidth (~300-600 KB per request):
  - 1,000,000 note ciphertexts is a 580MB database (currently there are ~400k).
  - That means a wallet can make ~1000 requests before it's cheaper to download the whole DB.
    - 1-minute intervals: 16 hours, 5-minute intervals: 83 hours, 10-minute: 166 hours.
    - 10 requests every time the wallet's opened: 100 wallet-openings.
  - Concrete bandwidth usage:
    - 1-minute intervals: 30MB/hour, 5-minute intervals: 6MB/hour, 10-minute: 3MB/hour.
    - 10 requests every time the wallet's opened: 5MB per wallet-opening.
  - Note: A 10-minute interval means if the wallet gets 6 transactions all at once, it will take an hour to get all of the memos. *During that time, the wallet must continue mempool polling.*
- Latency-bandwidth tradeoff can be a user-configurable setting.

Action Item #8:

Prototype a PIR implementation with a user-configurable privacy setting.

Confirm the bandwidth estimates, figure out how inserting records into the database will work, estimate server costs to support N users.

Action Item #9:

Find out what users’ privacy needs really are.

Bandwidth vs. robust privacy vs. quick memos?

Nation states vs. CipherTrace vs. criminals?

Configurable?

5. Everything else I considered...

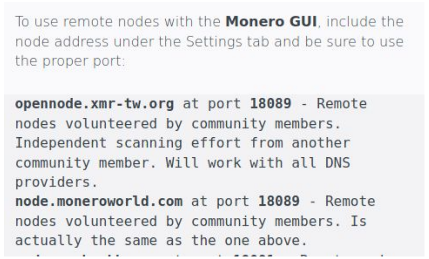
- Discouraging memo retrieval through UX.
    - We shouldn’t do this, memos are an important reason projects choose to build on Zcash.
- Ricochet & Cwtch.
    - These are metadata-resistant messaging protocols based on Tor hidden services. They don’t provide much value for our needs (assuming Tor is even feasible on mobile). In particular, they could worsen traffic-analysis attacks and wallets would somehow need to find each others’ Cwtch identifiers.
- Downloading the memo for every Sapling output.
    - This will increase the wallet’s bandwidth usage by 5-10x over the current protocol.
    - But in concrete terms, downloading 1M ciphertexts is only 580MB (currently there are ~400k).

Action Item #10:

Implement the download-all-memos defense.

What’s Monero doing?

- **No consideration of traffic-analysis attacks against wallets that I could find, even in academia.** Monero almost certainly has similar vulnerabilities.
    - https://ieeexplore.ieee.org/abstract/document/8845213
    - https://ieeexplore.ieee.org/abstract/document/8406557
    - These are papers that *should have* considered traffic analysis!
- View-key-protecting light wallets connect to a remote node and download all transactions to scan them.
- View-key-sharing light wallets *share their view key with a remote node*; the node learns which transactions belong to the wallet and sends only those.



To use remote nodes with the **Monero GUI**, include the node address under the Settings tab and be sure to use the proper port:

```
opennode.xmr-tw.org at port 18089 - Remote
nodes volunteered by community members.
Independent scanning effort from another
community member. Will work with all DNS
providers.
node.moneroworld.com at port 18089 - Remote
nodes volunteered by community members. Is
actually the same as the one above.
```

←From https://moneroworld.com/

Does this mean users are encouraged to share their view keys with random nodes!?!?

Action Item #11:

Make proof-of-concept traffic-analysis side-channel attacks against lighwalletd and Monero.

Action Items

- In priority order from top to bottom:
  - **#2 Research spike to investigate feasibility of integrated-in-app Tor on mobile.**
  - **#8 Prototype a PIR implementation with user-configurable privacy setting.**
  - **#3 Constant-bandwidth version of the current wallet protocol**.
  - **#11 PoC traffic-analysis side-channels against lighwalletd and Monero.**
    - This will help us confirm the fixes while generating novel research.
  - #9 Find out what users' privacy needs really are.
  - #4 Research approaches to hide outbound transactions broadcast by lightwalletd.
  - #5 Research spike to decide on note-merging / auto-shielding / etc. defenses to dust attacks.
  - #6 Continue long-term mixnet research for scalable-Zcash, aiming for mobile device compatibility.
  - #1 Validate block headers, proof of work, and the note commitment tree root.
  - #7 Research changes to liberated payments to help improve privacy for light wallets.
  - #10 Implement the download-all-memos defense.

Final Thoughts

To build scalable-Zcash, we’ll need to move the state of the art in anonymous communication networks. Let’s make sure we’re considering light wallets when we do that.

We can improve the current situation without waiting for a radical redesign, but there’s research to do it might make the wallet a bandwidth hog.

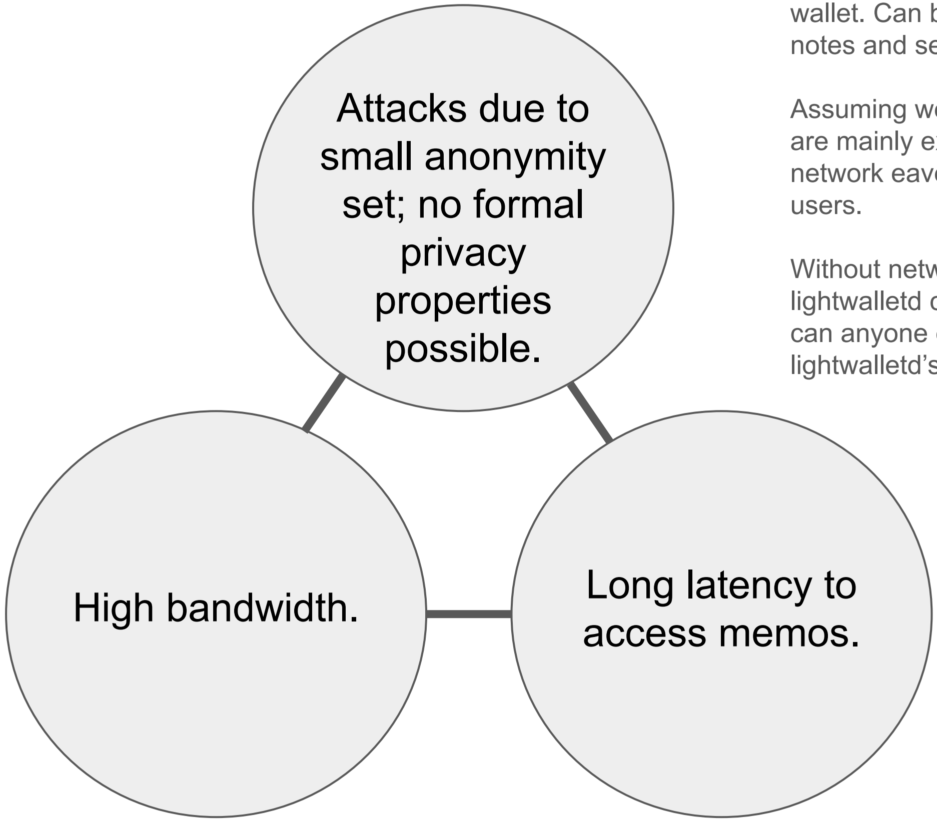
Traffic-analysis attacks on light wallets apparently haven’t been considered even in academic research. There’s an opportunity to do that research and educate the market.

Thanks to Bambam, Daira, Kevin, Larry, Linda, Pacu for discussion and feedback!

Slides I don’t need anymore but I’m too attached to to delete

The Tradeoff (pick one)

It’s possible to let the user decide using a configuration option.


Attacks due to small anonymity set; no formal privacy properties possible.
High bandwidth.
Long latency to access memos.

Aside from a small anonymity set of recent notes, the attacker learns which outputs *do not* belong to the wallet. Can be used to link users to notes and senders to receivers.

Assuming we use Tor, these attacks are mainly exploitable by passive network eavesdroppers close to the users.

Without network anonymity, evil lightwalletd can exploit them, and so can anyone eavesdropping on lightwalletd’s connection.