Recently I came across a proposal for safe zero-conf transactions on Bitcoin-style chains.

Basically it's a way to post a bond, such that a doublespend results in the loss of the bond. It detects doublespends by reuse of UTXO inputs. Anyone (presumably miners) can submit the two transactions and collect the bond.

We could build this into a Plasma chain, letting the operators collect the bonds. But by tweaking the idea we can do something similar on the main chain today.

Instead of UTXOs, we just look for reuse of the same nonce from the same sender. We can write a contract that accepts two complete signed transactions, checks their senders and nonces, and if they match, awards a bond to msg.sender.

The merchant has to do a few simple checks to accept a zeroconf transaction:

- The sender must have adequate funds
- The transaction must have the sender's current nonce, not a later one
- The sender cannot be a contract, since a multisig wallet could allow a different address to withdraw the funds
- The sender must have a deposit in the zeroconf contract, which is larger than the purchase amount

If all of these things are true, the merchant can safely accept a zeroconf payment. The transaction can be faster than a credit card approval if the merchant's client maintains the necessary state: just the current balance and nonce of all addresses, and all the current zeroconf deposits. Then it's a local lookup to approve the transaction. In a busy shop, this would be much better than waiting for a block, even with Ethereum's fast blocks.

The merchant can monitor for doublespends and attempt to collect bonds, in case the miners fail to do it. But if the miners find it worthwhile, the merchant can simply enjoy fast transactions with low fraud rates and not worry about it.

Implementing the zeroconf contract would be reasonably simple. There are Solidity libraries for RLP but we don't need them; the nonce is the first data field (aside from length indicators) and the signature is at the end. On each transaction check signature and nonce, and if the transactions match, award their signer's bond to msg.sender.

Obviously this doesn't work so easily with signature abstraction, but if the existing transaction format is still supported, the merchant can simply restrict zeroconf approvals to old-style transactions.