A somewhat overlooked way to scale small transfers is probabilistic payments. To pay you ten cents, I can give you a ticket with a 1% chance of paying ten dollars, and it only goes on chain if it pays out.

To make this work, we need a good source of randomness, and protection against replays. One method is to use block hashes and target particular blocks, though this will require us to record blockhashes in a more persistent way. If we assume users post a “current” hash to a contract, indexed by the hash of their email address, then we can use a hash onion per recipient. The sender signs sha3(receiver, targetHash, randomNumber) where targetHash is the current hash. The receiver takes the preimage of the targetHash, hashes it with the sender’s random number, and if the result is below a threshold then he submits the signed message and preimage to the contract and gets paid. The contract sets the preimage as the new “current” hash.

Vitalik posted in 2014 that it’s easy to doublespend against something like this, and suggested a large penalty bond, payable if the sender’s funds are insufficient upon redemption. We can do without that for some applications, like voluntary tips. In that situation a doublespend isn’t necessarily malicious; if the tip has a 1% chance of paying off, the user can reasonably deposit only enough funds to pay off a single tip while sending two concurrently, ignoring the 1/10,000 chance that both pay out.

For spam protection, we can make the doublespend problem go away entirely. Clients can require a probabilistic payment from unknown senders; let’s assume a penny per email makes spam unprofitable, and use a 1/1000 chance of getting $10.

A spammer may attempt to get around this by depositing only $10 and sending a million emails referencing that deposit. Naive recipients would have no way to tell.

But there’s an easy way to counter this: have the email client attempt to redeem the payment immediately, but wait an hour before showing the unsolicited email to the user. If most recipients do this, the spammer’s deposit is likely to go to zero in that hour. In that case, discard the email. This means the only person who reads the email is the one who got paid $10 to do so.

There’s no happy medium that helps the spammer. Even if he deposits $10,000, there’s an almost 50% chance that too many redemption attempts will succeed; then he reaches only a thousand people, paying $10 for each. To be confident of reaching a large number of people he needs a deposit large enough that it’s unlikely to run out. Thus the best he can do is pay an expected penny per email.