Blinded Bearer Certificates: a case for implementation

theymos, June 6, 2018

Blinded bearer certificates are useful whenever you need anonymity and/or extremely cheap transactions (for eg. microtransactions). I can easily imagine a world in which a bb-cert client is integrated in every Web browser and used on many sites, since the technology has the potential for widespread usefulness.

Example use-cases

Bitcointalk.org CAPTCHA bypass

Some users find it difficult to solve reCAPTCHAs on bitcointalk.org (on registration, login, and sometimes as a Cloudflare challenge) and other sites. Depending on your IP address, reCAPTCHA can completely ban you, or it can give you a series of challenges which take many minutes to solve.

As an optional alternative to solving the captcha, you could pay a very tiny amount, something like $0.001. However, block-chain-based payments systems will never work reasonably with such small amounts. Lightning and Mimblewimble are possible long-term solutions, but neither is usable now, and Lightning in particular may, like Bitcoin, be less-than-ideally private.

With blinded bearer certificates:

1. A user pays $1 via Bitcoin or similar.
2. The server issues 1000 CAPTCHA-bypass certs to the user.
3. Whenever the user comes across a CAPTCHA, they can instead click a “use cert instead” link and copy/paste one of their certs. Then they are allowed to continue without filling out the CAPTCHA, and that cert is burned.
4. The user can also transfer the certs to other people (eg. friends, or to people who paid using a payment method unsupported by the service) anonymously.

This uses mainly the microtransaction-friendliness of blinded bearer certificates rather than the anonymity.

I would be very interested in adding this to bitcointalk.org, but it also has potential as a new standalone, general-purpose service. One could set up a service which basically mirrors the reCAPTCHA API, but also gives end-users the option of bypassing the CAPTCHA with a unified CAPTCHA-bypass cert valid across multiple sites. Because the certs are anonymous, users would not need to worry about being tracked based on their CAPTCHA-bypass certs, which would be a major concern if a service like this just maintained a *balance* for each end-user instead of using bb-certs. A service like this could be marketed as a “Tor/VPN/privacy-friendly reCAPTCHA”, and could gain popularity in the privacy-aware Web niche.

Account credit, premium accounts, etc.

Let's say that you want to buy a domain name anonymously. Currently the best you can do is probably to pay Bitcoin on Namecheap. But Bitcoin is very much less than ideal when it comes to anonymity, and Bitcoin transactions can also be expensive, so if you wanted to buy 10 domains on 10 *different* Namecheap accounts in order to isolate them, the fees associated with these transactions could become significant. (Currently Bitcoin transaction fees are low, but this is not long-term-stable, and they will rise again, at least for on-chain transactions.)

Solution: sell "gift card" blinded bearer certificates in denominations of $10, $25, etc. You can buy them with a non-anonymous method, and you can even buy eg. 10 $10 gift cards in one single transaction, but the resulting certs are 100% anonymous and free-to-transact. Mechanically, it's similar to the above CAPTCHA use-case. As with that, you could transfer the certs to friends or sell them for an otherwise-unsupported payment method.

This would be very useful on all privacy-sensitive paid services such as Protonmail, VPNs, etc.

Paxful reputation

One of the more anonymous ways of buying bitcoins currently is to buy a gift card with cash and then sell it for BTC on Paxful. However, it is difficult to do this with a brand new Paxful account because BTC-sellers are wary of various types of fraud in this process (eg. buying the gift card with a stolen credit card); however, if you build reputation on a Paxful account, you will then be tying all of your acquired BTC together, which is terrible for privacy. This can be solved with bb-certs:

1. Paxful designates certain users as trusted. These users might, for example, have their identity known to Paxful.
2. Without having to create an account, a user can post an advertisement like "Selling a $100 Amazon gift card for BTC". When they do this, they will spend all of the Paxful trust certs that they have acquired at any time in the past. Their advertisement will then be listed with a trust score equal to the number of submitted certs.
3.
    a. If the advertisement is cancelled, then Paxful will reissue new certs equal to the previously-spent certs and give them back to the advertiser, so the advertiser's trust score will not change.
    b. If the advertisement is completed successfully, and the counterparty is a Paxful-trusted user, then Paxful will reissue new certs equal to the previously-spent certs plus 1. So the advertiser's trust score will either increase, if they are dealing with a Paxful-trusted-user, or stay the same if they're dealing with any other user.
    c. If the advertisement results in fraud by the advertiser, then no new certs will be reissued, and the advertiser will lose all of the trust points that he associated with the advertisement.

The result is that you can do Paxful trades without any possibility of your transactions being linked together, but *with* tracking of your reputation so that once you've done a few trades, you will be able to easily trade with people.

ChipMixer

Chipmixer.com is one of the more popular and theoretically-sound Bitcoin mixers. It works by splitting deposits into fixed-size "chips", eg. if you deposit 0.112 BTC then you will receive a 0.064 chip, a 0.032 chip, and a 0.016 chip, which you can then withdraw. This prevents value-based analysis of the mixer.

However, from the perspective of the mixer, there is still a clear link between the deposit and the withdrawn chips. bb-certs can prevent this:

- You deposit 0.112 BTC to Chipmixer
- Chipmixer issues you a 0.064 cert, a 0.032 cert, and a 0.016 cert.
- Later, you spend some or all of those certs on Chipmixer, telling them where to send the associated BTC.

This removes the link between your deposit and your withdrawal, even from the perspective of Chipmixer, since bb-certs are completely untraceable. Furthermore, Chipmixer could allow users to transfer the certs, which would allow for 100% anonymous and cheap off-chain transactions.

(This is a very "classical" usage of bb-certs, and ChipMixer is already arranged in a very convenient way for it.)

Existing implementations

Even though blinded bearer certificates have been well-known for decades, there have been few usable implementations. As far as I know, the main one currently is Lucre. However, Lucre has never been widely-used, it has not been updated in many years, and I am not sure whether its theoretical basis is still considered optimal. It's also under an annoying 4-clause-BSD-style license.

Open Transactions, which uses Lucre for its blind signing, is an attempt to create an all-purpose blinded-bearer-certificates protocol. However, despite existing for almost as long as Bitcoin itself, it has as far as I know never been used in any real-world project because its API is so complex and obtuse that nobody but its creator really understands it. More recently, Open Transactions has gone down a route of crazy ideas involving BitMessage which will never work. Due to all of this, it's probably best to ignore Open Transactions entirely.

Requirements of a basic implementation

A basic implementation would have two pieces: a client component that would run on the end-user's computer, perhaps in their browser, and a server component. Communication between them could be done via HTTP, or fairly easily via copy/pasting in a similar way to OpenPGP.

The client component should **not** merely exist as JavaScript provided on a web page, as this is insecure; rather, it should be a stand-alone browser extension or desktop/mobile application.

The typical procedures would be:

- The server creates a new denomination (eg. a $10 gift card and a $25 gift card).
- The server issues one or more new certs:
    - The client submits *y* unsigned certs to the server.
    - The server requests $*x* for *y* certs.
    - The client pays, and the server confirms payment.
    - The server signs and delivers the *y* certs to the client.
    - The client saves the *y* certs.
- The client redeems one or more certs (eg. uses a $10 gift card to add to an account balance):
    - The client sends one or more previously-issued certs to the server.
    - The server verifies the certs, and if they're all valid, they trigger whatever payment-registered action is appropriate.
    - (Optional) If necessary, the server may issue one or more new certs as *change*. This exchange would require that the client present a new unsigned cert, etc.
- The client transfers one or more certs:
    - Alice wants to transfer certs to Bob. She just sends him the previously-issued certs, possibly encrypted to prevent eavesdropping. Note that transfers cannot be anonymously done by using the server as an intermediary, but rather some out-of-band communication method should be used (eg. email).
    - Bob redeems the certs with the server as above, but requests that they all be immediately reissued.
    - Only after the server reissues the certs, Bob can consider the payment complete.

Note that:

a. The server will actually involve coordination between three (or more) pieces of server software: a Web server talking to the client, a payment server checking the non-cert payment (eg. bitcoind), and the cert-server handling the bb-cert stuff.
b. The cert-server is highly security-sensitive. If it is compromised, all issued certs are immediately busted. So you should expect it to typically be on a separate machine from the Web server.

For all of my example use-cases, this basic implementation is sufficient.

More advanced implementations

The basic implementation can be made more decentralized and secure in at least these ways:

First, the cert-server can actually be a multisig arrangement among multiple servers, each of which may be controlled by independent entities. In order for a cert to be considered valid, it would have to be signed by *n* of *m* of the servers' keys. This prevents one compromised machine from ruining all issued certs.

Second, you can have two layers of cert-servers: “offline” and “online”. The offline servers are air-gapped and only synced-up occasionally, whereas the online servers perform all of the usual duties of a cert-server. If the online cert-server(s) are compromised, then only the certs issued since the last offline-server-sync-up would be invalidated, allowing the system as a whole to survive.

Third, in cases where the cert represents an underlying asset which itself supports decentralized smart contracts (such as Bitcoin), then the above multisig arrangement can be enforced by the asset itself.

A full off-chain system for Bitcoin transactions would combine all of the above improvements:

- The bitcoins would actually be held in an n_1-of-m_1 multisig smart contract managed by m_1 offline cert-servers.
- A different set of m_2 online cert-servers would have a different n_2-of-m_2 multisig arrangement amongst themselves. This multisig would be purely in the bb-cert system, and would not exist on the Bitcoin block chain. These servers would handle day-to-day transactions.
- Periodically (daily/weekly/monthly), the offline cert-servers would sync up with the online cert-servers.
- Withdrawals would not result in actual BTC withdrawals until the next sync-up.

This has equal or better security compared to a federated sidechain such as Rootstock, but it is perfectly anonymous and more efficient. As with federated sidechains, if the offline servers are owned by several totally-independent entities, then the level of decentralization, while undeniably far worse than on-chain-Bitcoin, may be good enough to be acceptable as a replacement for most day-to-day on-chain Bitcoin transactions.

Again: these advanced features are not required for many significant use-cases, and I mainly envision them as part of a possible alternative to eg. Lightning. A standard business which wants to use bb-certs is not going to be running 12 offline servers and 20 online servers spread across the globe; in almost all use-cases by *businesses*, they will want and need just one cert-server.