CheckBits

Ada Taylor, Brendan Chang, Erica Du, Parker Tew

December 9, 2014

Abstract

Bitcoin, while originally designed for digital spaces, is gaining enough mainstream popularity to be increasingly offered as a method of payment in the physical world. However, for day to day use of Bitcoin in physical point of sale applications there are two problems. The first is double spending, where a customer issues two checks which are both valid individually but if both cashed would overextend the customer's wallet, creating an effect analogous to a bounced check. Merchants do not have a guarantee that the customer is not double-spending until their payment is confirmed, which can take at least 10 minutes. The second problem is that because the blockchain is public, users leave trails to their primary accounts and other spending patterns with every transaction. This can give away information about their assets, spending patterns, contacts, and purchases that many users generally wish to protect.

We would like to propose a guarantor system which solves both of these problems. Instead of paying for services directly, the customer will pool their money with others in a trusted guarantor. They will then inform the guarantor with a signed and encrypted "check" when they would like to make a purchase. The guarantor makes the payment with Bitcoin on their behalf, leaving it ambiguous which of the users of the guarantor made the purchase. The guarantor then posts a signed notification of the transaction's validity on a public bulletin, where the merchant can check to get an immediate response from the guarantor that the unconfirmed transaction on the behalf of the customer will be valid.

This systems allows for a wide variety of guarantors, from credit and debit systems to more trusting models that might be appropriate for small living groups.

1 Threat Model

We assume that the attacker has complete access to the blockchain's record, which is publicly accessible. We also assume that they have access to all traffic, though they are unable to decrypt without a key, which the attacker does not have access to.

2 Approach Overview

2.1 Guarantor

Our system introduces a middleman, the guarantor, which pools the Bitcoin of multiple clients together, so that a single charge cannot be correlated with any specific user of the guarantor by looking at the blockchain. The guarantor's first responsibility is to accept money and establishing a secure relationship with the customer with an exchange of keys. Its second responsibility is to make purchases on the behalf of its clients in response to their signed and encrypted requests, using the Bitcoin pool it controls.

There is a large degree of flexibility for the guarantor outside of these parameters. A service could be run on a debit model, where users are only allowed to spend up to the amount that they stored initially in the guarantor. Alternatively, the guarantor could work as a credit system, where the guarantor retains a small amount of information about each customer for billing purposes, and allows them to spend and then pay afterwards. A third possibility would be payments with a slight percentage paid to cover future overcharges and fraud. Guarantors are incentivized to exist by charging a slight fee for transactions. While each of these models may have its own security considerations, we are not concerned with these explicitly for the purposes of this paper.

Scale is also variable. While a guarantor could provide the service for hundreds of users, a small group of trusted friends could also establish a small guarantor for their use. This guarantor could even be a client of a larger guarantor, if the customers wanted their checks to have the authority of a larger and more well known guarantor, but have an additional layer of privacy.

2.2 Bulletin

In order to provide the customer with a nod of approval from the guarantor immediately, without waiting for the blockchain to stabilize and confirm the transaction, we also introduce a bulletin service. This service is a place specified by the merchant that it will check for notifications verifying valid or invalid payment. These notifications provide the transaction id and the status, and are signed by the guarantor. This system allows the merchant to check one location for that data while are still allowed flexibility in their choice of guarantor. It also does not require that the merchant be burdened by hosting a server just to receive notification of payments.

The bulletin is not responsible for the integrity of the checks, or their contents, and all information provided there is public. However, these notifications are signed by the guarantor, which makes them extremely difficult to spoof.

2.3 Backwards Compatibility

It is of note that our system is designed to interact both others using our system, or those making standard Bitcoin transactions. Should a merchant only have a Bitcoin address, and not be prepared to use our system, the guarantor still makes a payment to that address, though they do not gain the benefit of an immediate decision on whether they think the payment will be valid. This also is the use case for if the guarantor is completely untrusted by the merchant - they can simply accept the transaction at face value as they would a normal Bitcoin transaction, or wait 10 minutes before trusting it.

The customer also retains their privacy if the merchant is simply providing a Bitcoin address, because that does not affect the privacy that the pooling within the guarantor provides. The transaction remains mediated by the guarantor regardless.

3 Anatomy of a purchase

There are two phases to the system: the customer establishing their relationship with the guarantor, and the customer sending money to a merchant via the guarantor.

3.1 Setup

In the first phase, the customer transmits money to the guarantor, and the guarantor gives back to them an encryption key or "checkbook". This key will be used to verify the identity of the customer, and in encrypted communication with the guarantor.

Care should be taken to choose an encryption scheme for this checkbook which does not allow an adversary to associate checks created by the same customer.

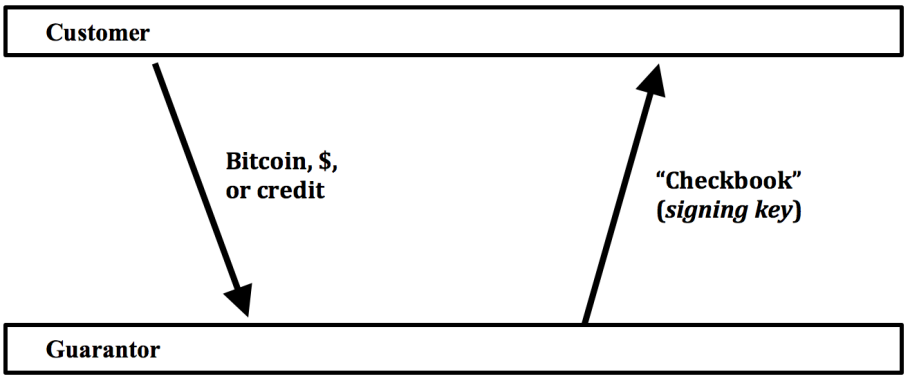

Figure 1: Initial setup

3.2 Payment

Before payment, the merchant provides a Bitcoin address for the customer, transaction id, amount, and preferred bulletin. This could be provided via QR code, or even non-digitally on paper or a sign. This is

the only direct point of interaction between the customer and the merchant, and can be provided in the same manner as the good or service which is being paid for. Note that none of this information provides information about the identity of the customer.

In order to make a payment, the customer uses their "checkbook" to sign and encrypt their purchase information for the guarantor: the amount, receiving address, transaction id provided by the merchant, and a timestamp. If the customer uses TOR to send this check to the guarantor and the encryption scheme chosen does not allow association of checks written by the same author, there is no way for an attacker to associate the identity of the customer with the payment.

Upon receiving the payment, the guarantor decrypts the check and verifies the signature of the customer. They then make the payment to the specified address, and post to the specified bulletin a signed message of the transaction id, status, and timestamp.

The merchant can then query the bulletin to see that the payment is incoming and vouched for by the signing guarantor.

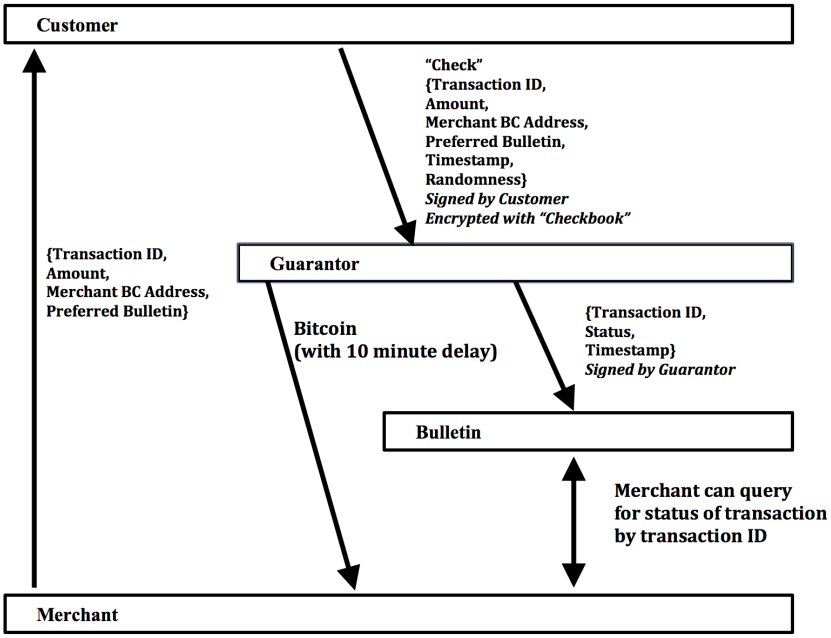

Figure 2: A sample transaction

4 Security Provided

4.1 Guarantor

Because all transactions move through the guarantor, the client never leaves a direct Bitcoin trail of transactions to the merchant. An observer of the Bitcoin transaction log cannot easily associate customers with transactions, because any of the clients who have used the guarantor in the past could have authorized the payment. Therefore, the more people using the system the harder it is to attempt to track history.

Timing attacks are mitigated primarily because the checks cannot be associated with each other or (if they are using TOR) a given user. Therefore, while a check can be associated with a payment, the check cannot be associated with a specific individual.

The guarantor may decline to honor two checks with the same id and timestamp fields in a short period at their discretion, in order to prevent replay attacks.

This system is still vulnerable to statistical attacks, but the more users and transactions handled by the system, the more difficult this becomes. Additionally, while smaller systems may be easier to attack, they are also lower value targets, which would in itself deter attackers.

4.2 Bulletin

The bulletin service can be untrusted, since every notification posted is signed by the guarantor and that information is bundled to avoid tampering with individual elements. Merchants only have to trust the guarantor in order to take payments immediately rather than waiting for the Bitcoin transaction to complete. Again, if the guarantor is unknown or untrusted the merchant can ignore the bulletin board notification completely, and just wait for a Bitcoin transaction as normal.

The only information an outside observer can glean from the bulletin board is how many payments the guarantor is making and when (but not how much). They could then visit the guarantor's Bitcoin pool and see that merchants are having payments given to them via this system- but this is true of the current system, as well.

5 Our specific implementation

While there is a great deal of flexibility in the role of the guarantor, we have chosen to implement ours on a debit system accepting BitCoin transactions. Admittedly, this is motivated in part by low financial risk on the part of the guarantor, but we also aim to demonstrate the differences between our system and vanilla Bitcoin transactions, as well as provide an example of one of the most likely schemes to chose for small living group use.

In our specific example, the user creates an account with the guarantor. They can then provide the guarantor with their public key, and this allows the guarantor to verify that checks are actually from them. The client will provide the guarantor with a key such that the customer can encrypt messages to them and have them decrypted by the guarantor.

The guarantor also provides the customer with a randomized address to its centralized wallet. This address is one-time use and allows the guarantor to determine which customer deposited value without relying on knowledge of which customer claims they own a particular wallet or transaction. The customer can ask for a new address at will.

This means that if a user already has two previously existing completely unassociated wallets, they will not betray to an outside observer the fact that they own both if they make deposits through two randomized deposit addresses. Wallet addresses do not contain any unique information about the customer they belong to, and are randomly generated in their entirety.

Therefore, the balance that the guarantor contains depends only on selected deposits that the customer wants to place, using the secret knowledge they have from their account with the guarantor. An attacker with control of a past or current wallet cannot compromise the relationship between the guarantor and the customer, and merely claiming to have a given wallet address does not provide any access or additional knowledge.