BLIND SIGNATURES FOR BITCOIN TRANSACTION ANONYMITY

WATSON LADD

Abstract. Bitcoin is [6] a peer-to-peer distributed currency system. Bitcoin attempts to provide anonymity to users by cycling keys [6] however such protection is of limited value. [8] In this paper I present a new means of forming transactions that prevents a sender of bitcoins from linking the public key of the reciever to a transaction, requiring only a single additional option for what data is signed.

1. The Problem

Let Alice pay Brian, Beth, Bob, and Brent. Brian, Beth, Bob, and Brent want it to be impossible for Alice to link a recipient of the transactions she signs to one of them. While she must know $\{Brent, Brian, Beth, Bob\}$ and cannot pay them different amounts without trivially linking the payments, it is possible for her to not know the mapping between the set of people she payed and Bitcoin addresses recieving payment.

2. The structure of Bitcoin

Much of this is from [6]. Bitcoin uses a globally known list of transactions, each transaction being part of a block. Blocks contain a proof of work, along with a hash of all previous blocks. A transaction has inputs and outputs,

Each coin is a list of transactions starting from the first owner of a block, who mints the coin.

A transaction has inputs and outputs. The inputs are the outputs of previous transactions, along with information enabling redemption as desribed in the next section.

Bitcoin defends against double-spending by requiring a transaction to appear in the global chain to be valid. The finder of a block is only rewarded with Bitcoins if all transactions in the block are valid and do not result in double-spending.

The global chain poses challenges for anonymity. It is possible to rotate keys to attempt to delink transactions, however as discussed in [8] in practice this does not ensure anonymity. Tracking the flow of funds between keys offers clues as to which keys are controlled by the same person, and so link different transactions together.

2.1. The Bitcoin Transaction Format.

A Bitcoin transaction links inputs to outputs. Each input and output consists of a value and a script. The script imposes

Date: March 4, 2012.

2010 *Mathematics Subject Classification.* 94A60.

Key words and phrases. electronic cash, Bitcoin, blind signatures.

a condition on any transaction using it as an input, namely that values must be provided such that the script validates the transaction.

Scripts are sequences of instructions for a stack autonomon. Scripts are given inputs specified in the output half of the transaction, and are then run on the autonomon. Success is indicated by the last instruction terminating with a true value on the stack.

Scripts are capable of stack manipulations, conditional execution, comparing data on the stack, hashing, and validating signatures on the entire transaction or just subparts of the transaction. Such partial validations are useful for various kinds of contract. The extent of validation is set by the provider of the signature: validation can include the outputs of the script or not.

Scripts can control what portions must be signed by use of the OP_CODESEPERATOR opcode. Execution of this opcode causes the prior portion of the script to be forgotten when OP_CHECKSIG is executed.

The signatures are standard ECDSA signatures.

3. Necessary Modifications to Bitcoin

We need a new opcode OP_CHECKHASHSIG, which given a signature, a public key, and a hash confirms that the the signature is a signature of the hash under the public key.

We also require a new signature type that blanks the index of the ouput of a transaction being used as an input before signing. Note that in the existing Bitcoin system identical outputs do not occur: this signature type, SIG_FUNGIBLE, stil protects the output script, so outputs must be carefully crafted to be indistinguisable with OP_CODESEPERATOR

Neverthless this new signature type should be generated sparingly. It is however essential to preserving anonymity: it enables multiple signatures to be made that can grant access to indistinguishable pools of funds, while securing the recipient from theft by those to whom the blockchain is sent.

4. The Protocol

4.1. **Setup.** Alice wishes to pay a total of T evenly divided among n participants $B_1, \dots B_n$. n is limited by the maximum size of a script in a transaction. Alice generates an ECDSA public key $A = x_a P$.

Alice generates a single transactions with n outputs, each of value T/n and with a value script demanding i distinct signatures with key A: $i-1$ of the signatures are of hashes given by the redemer, and 1 is a signature of the script. Each script has the same hash via the use of OP_CODESEPERATOR, meaning that a SIG_FUNGABLE signature of one of the outputs is a SIG_FUNGABLE singature of any of the outputs.

4.2. **Paying.** Alice must pay B by signing a transaction B designed of a specified form, without knowing what transaction was actually signed. This is achieved by cut-and-choose in conjunction with blind signatures.

We use the protocol of [5] modified to include cut-and-choose. We make use of the zero-knowledge proofs discussed in [2] and the Paillier cryptosystem of [7].

B begins by picking a Paillier scheme public key m of size between q^6 and q^7 and transmitting it to Alice. He also computes hashes $H_1, \dots H_k$ of k transactions,

each using an output of the transaction generated in step 1 to pay Bitcoins to an ECDSA public key that B wants to transfer Bitcoins to.

B has of course generated these public keys anew for each transaction.

B transmits to Alice commitments as in [2] to each H_i. Alice selects an integer i between 1 and k and has B reveal all the other messages along with sufficent information to determine the validty of the commitments. If B cheats, this will be detected now.

Alice now selects a random k_a in the range $[1, q-1]$ and transmits $k_a P$ to B. B selects a random k_b and computes $k_b(k_a P)$. Let r be the x-coordinate of that point, and let $z_a = 1/k_a$, $z_b = 1/k_b$.

B sends to Alice $a = E((rz_b) \mod q)$, and $b = E(H_i z_b \mod q)$ along with a proof as in [2] that a and b are Paillier encryptions of integers less then q and greater then 1.

The Paillier system is additively homomorphic and permits efficient multiplation of plaintexts by constants. Multiplying ciphertexts is addition of the plaintext values, and exponentiation is multiplication by a constant.

Alice computes $c = a^{x_a z_a} b^{z_a} E(dq)$ where d is a random integer in the range $[1, q^5]$. B decrypts c to obtain s, the other half of the signature, after taking it modulo q. As the public key is large enough to prevent overflow this gives the correct answer.

This scheme is a secure blind signature scheme as Alice learns nothing about the value of H_i by the semantic security of the Paillier scheme. B learns nothing about Alice's private key beyond the signature thanks to the addition of dq to the result.

4.3. **Redemption.** All B's now have their signatures. To redeem a signature B simply pushes the transaction that was signed to the blockchain, selecting as output the first unused output and giving it the previously used signatures and hashes of the associated transactions, along with the until now unseen new signature for the transaction.

Alice could attempt to swindle some of the B's by issuing more signatures then there are outputs or redeeming outputs herself, then seeing who complains about not getting paid. But the complainant has ample proof of Alice's trechery: there are more signatures with the private key used for the transaction then there are outputs. So long as this is not the case, payment goes through. The ability to reveal that Alice is a cheat with proof thereof is sufficient deterrent to fraud on Alice's part.

Alice cannot determine the mapping between the public keys to which the Bitcoins were transfered and the identities of the B's because she handed out the signatures blindly. Even if the temporary identies are linked to the B's permanant identies Alice remains in the dark: the revealed keys were never used and so remain unlinked to identities.

As a result the specific transaction Alice engaged in remains unlinked from the identity of the B's. This delinks the source of funds from their eventual use.

5. Practicality and Further Work

The anonymity set is limited to all persons whom Alice pays using the same transaction, and hence by limitations in the current Bitcoin implementation on the size of transactions and their scripts. These limitations exist to limit the cost of validating a block. [1]

This present scheme is neverthless useful. Many users of bitcoins obtain them through exchanging USD for Bitcoins, a transaction that inevitably links identifying information to the recipient address. The use of this protocol would prevent that association from being made, provided the anonymity sets are large enough.

Further research is required on increasing the anonymity set, as well as methods for anonymizing the senders of funds. Multiparty privacy-preserving set unions as in [4] could be used for increasing the anonymity set, but require significantly more communication and computation then the method presented here. They can also be potentially extended to anonymize transactions of different face values: each participant demonstrates that they will commit an input and an output of the same face value to the union. They also require coordination amongst all recipients of bitcoins in the anonymity set.

The limitations on the anonymity set that are most tight are on the size of validation scripts. A script that determines that n arguments are distinct requires size $O(n^2)$ due to the deliberate absence of looping operations. Loops of bounded cost would be an addition that would ameliorate this constraint, as would permitting a search through the blockchain to see that a prior transaction was spent. The second option is dangerous: scripts must be monotonic, and searching through the blockchain is a potentially non-monotonic operation that could open the door to double spending.

It may be possible to extend this method to providing a Bitcoin-based digital cash scheme where double-spending would be prevented by revealing information required to redeem a penalty. Such schemes were first discussed in [3]. Such a scheme would not depend on a central party to restrict the amount of cash created, provide very strong anonymity, and could be used to pay recipients willing to take Bitcoins. It would however require significantly more changes to Bitcoin then the current proposal, and would have to deal with the problem of having the ultimate recipient of funds be unnamed.

References

[1] *Satochi's Client*. `https://github.com/bitcoin`.

[2] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology EUROCRYPT 2000*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[3] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology CRYPTO 88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Berlin / Heidelberg, 1990.

[4] Lea Kissner and Dawn Song. Privacy-preserving set operations. Technical Report CMU-CS-05-113, Carnegie Mellon University, February 2005. `http://www.cs.cmu.edu/~leak/papers/set-tech-full.pdf`.

[5] An Metet. Blind signatures with dsa/ecdsa?, 2004. `http://lists.virus.org/cryptography-0404/msg00149.html`.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. `http://bitcoin.org/bitcoin.pdf`.

[7] Pascal Paillier. Public-key cryptosystems based on composite degree residuoisity classes. In *Advances in Cryptology EUROCRYPT 1999*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[8] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. 2011. `http://arxiv.org/pdf/1107.4524v1.pdf`.

E-mail address: `watsonbladd@gmail.com`