Combining Bitcoin and the Ripple to create a fast, scalable, decentralized, anonymous, low-trust payment network (draft 2)

January 15, 2013

Contents

Chapter 1

Introduction

1.1 Bitcoin

Bitcoin uses a proof-of-work chain, called the *block chain*, to create a distributed, fully decentralized timestamp server. The objects that are being timestamped are transactions; the validity of transactions and their consistency with previously timestamped transactions are checked: transactions that fail this test are not included in the block chain. Because of this, the block chain acts as a decentralized consensus making device. For instance, in the case that certain bitcoins are spent twice: the two transactions are inconsistent with each other, so eventually only one will end up in the block chain.

The downside is that it takes a relatively long time before a reliable consensus is reached: on average, a new block is created once every ten minutes, but since it is possible that in the end a block is not used in the block chain, it is necessary to wait for several blocks to make sure the transaction is irreversible.

Also, the solution scales very poorly with the number of transactions: to verify a single Bitcoin transaction, one needs acces to *all* Bitcoin transactions, at least from the moment of the oldest "input" of the transaction.

1.2 The Ripple

The Ripple monetary system consists of a decentralized peer-to-peer network. The nodes of the network are individual users, and the connections between nodes are credit relationships between users. Transactions are performed by finding a payment route between payer and payee, and letting each intermediate party receive money from the payer side, and send money to the payee side of the route. Every transaction changes the credit balance on the credit relationships that are involved, such that the total balance for intermediate parties remains unchanged[1], the total balance for the payer decreases and for the payee increases.

[1] except for a possible transaction fee, which will typically increase the balance of the intermediate party, as a reward for assisting in the transaction.

Since the Ripple works with credit relationships, neighboring parties (who have a credit relationship) need to trust each other. Preferably, they know each other's identity, so that payback of debt can be enforced by law, if necessary. The Ripple provides neighbors strong evidence for the amount of debt they have to each other.

The following phases can be identified in a Ripple transaction:

- Route establishment: a route through the network, between payer and payee, is found.
- Promising: neighbors in the route promise to pay each other (pay = change the credit balance) under the condition that cryptographic evidence is provided that the transaction is going to be committed.
- Payment: The cryptographic evidence of transaction commitment is distributed, and credit balances are changed according to the promises.

Depending on the exact concept to be chosen, these phases can be split into different passes, or one or more of these phases can be combined into a single pass.

The main problem in the Ripple is that transactions are not committed atomically. Depending on the exact details, this can have different consequences. If a part of the participants commits the transaction while another part aborts it, then some participants lose value while others benefit. This can often be exploited, such that a participant can let a transaction fail in such a way that he benefits and some other participant loses. This can be solved by having no abort/rollback mechanism. This removes the financial motivation for creating transaction failures, but a transaction that cannot be rolled back and will not be committed locks up money of participants ("nobody can spend it"). This is a serious Denial of Service.

The analysis of a detailed Ripple concept should not only focus on the "good flow" of a transaction and how it generates cryptographic proof for its participants, but also on the ability to abort a transaction in all its different stages, and the consequences when a participant no longer follows the "good flow" (intentionally or unintentionally).

1.3 Summary and properties of the proposed system

The payment system as proposed in this document is a variation on the Ripple. While the Ripple allows payments in any currency, this variation specializes on Bitcoin transactions. As a result of the way how Bitcoin is used, people need less trust in each other to become neighboring nodes. Because of this, more and higher-capacity links can be created, potentially even in the absence of law enforcement, e.g. between people who remain pseudonymous to each other. Also, the risk of denial of service attacks can be reduced.

To summarize the properties of the proposed system:

Speed: When there are no problems while performing a transaction, the time between starting the transaction and having it fully, irreversibly committed is limited only by communication delays between nodes. A transaction should typically take less than a second. When there are problems with a transaction, the worst-case delay until the final, irreversible decision between commit and rollback is limited by the time between Bitcoin blocks. Even a problematic transaction will be resolved within approximately one hour.

Scalability: Transactions without problems, which are expected to be most transactions, will not be registered in the Bitcoin block chain. As a result, the size of the block chain will be much smaller than in a pure Bitcoin system. Unlike some Ripple concepts, the selected routing mechanism allows for the use of routing tables, so the network can probaly reach a similar size as the Internet. Even then, the number of "uncertain" hops in a route is expected to be much smaller than on the internet, since most users will probably cluster together on a small number of hosts, and each host can solve the routing problem internally in a trivial way.

Decentralization: While it is allowed, and for routing purposes desired, that many users will share a single host computer with each other, users are not required to do this, and every user is allowed to set up his own host computer. Hosts form a decentralized peer-to-peer network, and as long as a person has at least a single payment connection with another user, that person can perform transactions with the entire network through that connection.

Anonymity: The identity of the payer and payee can be hidden from intermediate parties, and they can remain pseudonymous for each other. Direct neighbors in the network can remain pseudonymous for each other; parties which are not direct neighbors remain anonymous for each other. A hosting provider and its user can remain pseudonymous for each other. Measures can be taken to avoid that one neighbor gains knowledge about the contact (identity/pseudonym, balance) with another neighbor.

Low-trust: Users need to fully trust the host they use for using the system, but they can choose to run their own host. Neighbors are unable to steal from each other, but they can perform a denial of service on each other. When that happens, both neighbors are unable to access their part of the money, until the denial of service is resolved. Payer and payee need to trust each other (usually only in one direction) if delivery of the good or service is not inherently linked to performing the payment. No trust is needed between non-neighboring users.

Chapter 2

Ideas

2.1 Anonymous transaction routing

2.1.1 The communication network

The Ripple works on top of a communication network. Neighboring nodes in the Ripple network are expected to have a way of communicating with each other. Non-neighboring nodes may or may not have some way of communicating; in a transaction, payer and payee are expected to have some (temporary) communication channel between each other. The exact nature of these communication channels should not be specified, so that the Ripple network can be implemented on multiple types of communication channels. For instance, most nodes will simply communicate over TCP/IP+SSL, but some may choose to use the TOR network. Other may choose to use a dedicated physical cable. In POS systems, some combination of RFID+bluetooth or RFID+WLAN might be used. The only thing required of a communication channel is that it is sufficiently fast to finish transactions within their time-out limits.

2.1.2 Public key exchange

Two parties who communicate with each other might want to use a digital signature system to authenticate messages, and an encryption system to make sure that messages remain confidential. They need to have a verified public key of each other for this purpose.

In order to improve anonymity, a node can use multiple public keys:

- One permanent[1] public key per neighbor. Only you and that neighbor know that this public key belongs to you. In fact, knowledge about this public key is not useful for anyone else, because it will only be used for communication with this specific neighbor.

[1] Permanent, until there is some reason to replace the key. This is the same as e.g. for SSL certificates and PGP encryption.

- One temporary public key per transaction. For each transaction, a new key pair is created. For the neighbors involved in the transaction, the temporary public key is signed with the permanent key, so that, for the neighbors, a signature with the temporary key (within the context of the transaction) has the same value as a signature with the permanent key. This temporary key can be transmitted to all other parties in the transaction. Only your neighbors in the transaction can see that this key belongs to you; other participants in the transaction can not even see that you are the same person as in another transaction that was also processed by you.

In fact, instead of using a single temporary public key in a transaction, you might use multiple keys: one for each neighbor, and an arbitrary number of "intermediate" keys. To other participants, those public keys look the same as a chain of separate participants. By randomizing the number of keys to be used in a transaction, other participants can not be certain about the actual number of participants in the transaction. In fact, even if your two neighbors worked together, they would not be able to figure out that they have a shared contact.

2.1.3 Routing

Not exchanging any permanent identity information beyond direct neighbors creates a problem for routing transactions through the network. Some Ripple concepts deal with this by doing completely "blind" routing (trying all possible directions without knowing which direction is most likely to succeed).

There is another possibility: if some nodes in the payment network allow their permanent public keys to be published across the entire network, they can be included in routing tables. Payer and payee can then negotiate which of these "publicly known" nodes to use as a "meeting point". After agreeing on a meeting point, both can find a route to the meeting point, and the two routes together form the payment route. Payer and payee can both remain anonymous, while the "meeting point" node can remain pseudonymous. In fact, even the direct neighbors of a meeting point don't have to know that they are direct neighbors: they only know that their neighbor provides a route to the meeting point.

To some degree, this "meeting point" set-up is similar to how TOR hidden services are set up.

2.2 Using Bitcoin for low-trust "shared-property" Ripple accounts

2.2.1 Shared property instead of debt

One of the disadvantages of having a credit relationship between neighbors in the payment network is that it is possible for participants to have a lot more debt than they are capable of paying back. Even when neighbors know each other and can inspect each other's properties and income, the problem still exists, since a

participant can have debts to a large number of people, and those people have no way to know about any of the other debts. In the current financial system, this is solved to some degree by inspections of financial authorities, but the decentralized nature of the Ripple makes this impossible.

The proposed solution here is to work with shared property instead of debt. The idea is that every pair of neighbors has a certain amount of shared property, and they agree with each other which fraction of the shared property can be claimed by the one, and which fraction can be claimed by the other. As long as the shared property can not simultaneously be shared with third parties, the problem of "hidden debts" does not exist.

2.2.2 Using Bitcoin

The shared property concept can be implemented with bitcoins, by giving bitcoins a ScriptPubKey that requires signatures from both neighbors for withdrawing. If one of them wants to withdraw some of the shared bitcoins, he needs permission from the other to do so. Since M-of-N signature functionality is soon expected to be accepted as a "standard transaction" in the standard Bitcoin client, this concept does not require any new functionality in Bitcoin.

With this concept, the required trust between neighbors can be a lot lower than when using debt. In fact, it is not unthinkable that neighbors remain pseudonymous to each other. Neighbors can not "steal" from each other: the worst they can do to each other is a "denial of service", by refusing to sign the transaction when the other party wants to withdraw bitcoins from the shared account. This can be done in a "hijacking" set-up, where the malicious neighbor refuses to release any of his victim's money, unless the victim agrees to give a certain percentage of his money to his malicious neighbor. This is something to be taken into consideration when people decide to become neighbors in a pseudonymous way. As a counter-measure, they might for instance require that both parties will always own an approximately equal part of the shared bitcoins, so that in a denial of service scenario, both will lose access to a similar amount of money. This will discourage both parties to enter the denial of service scenario.

2.2.3 Sequence numbers and lock times

Another way to reduce the risk of "hijacking" is to use the (currently disabled) Bitcoin feature of sequence numbers and lock times. The idea is that the neighbors always have a Bitcoin transaction that sends each part of the shared account to its owner. Each time the balance between the neighbors changes (because of a Ripple transaction happening through the link), they create a new Bitcoin transaction that replaces the previous one. This Bitcoin transaction has the following properties:

- It has a nonzero lock time, somewhere in the future, and a sequence number that is less than UINT_MAX. Because of this, the transaction will not be included in the block chain until the lock time is reached.

- Each time the transaction is replaced, the new transaction has a sequence number that is higher than in the previous transaction. Since Bitcoin only accepts the transaction with the highest sequence number, this effectively makes the previous transaction invalid.
- Like in the previous section, each transaction requires signatures from both neighbors.

Now, if one of the neighbors wishes to withdraw money from the shared account, he can just create a new Bitcoin transaction that performs the withdrawal and (if applicable) sends the rest to a new shared account. If the other neighbor refuses to sign this, then the only problem is that the to-be-withdrawed money is not *immediately* available: it will automatically become available as soon as the lock time is reached.

A disadvantage of this approach is that the shared account can not live longer than the initially chosen lock time. It is not advisable to increase the lock time in later versions of the transaction: as soon as the initial lock time has passed, older versions of the transaction can be accepted into the block chain, and the only way to prevent this is to continuously make sure that Bitcoin miners are aware of the existence of the most recent version. So, the best way to deal with this is to switch to a new shared account, as soon as the old one is about to reach its lock time. Since this involves placing a Bitcoin transaction into the block chain, it should be done as infrequently as possible.

Also, this approach is insufficient to make the neighbor link completely trust-free. In fact, it might be less secure than the design of the previous section, because another exploit is possible: after a large Ripple transaction is committed, the "paying" side of the link can refuse to sign the updated version of the Bitcoin transaction. Then, all he has to do is wait for the lock time and make sure the last signed transaction is known to the Bitcoin miners: that one will give him the amount he had *before* paying, which is more than he should receive.

The only way to secure against this exploit, is to exchange and sign the new transaction version during the *promising* phase of a Ripple transaction, and making sure that the conditions of committing the Ripple transaction automatically makes the new Bitcoin transaction valid. This might be feasible if the commit state of a Ripple transaction can be determined with only some combination of digital signatures; unfortunately, the commit/rollback concept of section 2.4 seems too complicated to be used here.

2.3 How the Ripple improves speed and scalability of Bitcoin payments

Transactions that are performed through the Ripple network don't need to be registered in the Bitcoin block chain. If the Ripple concept becomes popular for Bitcoin payments, it is expected that a large majority of the transactions can be routed through it. The only major remaining use cases for traditional Bitcoin transactions are the following:

- Depositing / withdrawal from the shared accounts between neighbors
- Transfer of amounts of Bitcoins that are so large that the Ripple network can not handle them
- Transfer to/from traditional Bitcoin wallets, for e.g. large long-term savings, or other scenarios where trust in Ripple neighbors is insufficient
- The final commit/rollback mechanism, as explained in the next section

Because of this, the size of the Bitcoin block chain can remain much smaller than in a traditional Bitcoin economy. Keeping the block chain small assists in keeping Bitcoin decentralized, since it allows a larger number of people to have access to the full block chain.

2.4 Using Bitcoin for the final commit/rollback decision

2.4.1 Necessity of rollback

One of the problems in Ripple networks is that, whenever a participant does not respond as is expected of him for the "good flow" of a transaction, a transaction has to be rolled back, or else it will stay forever in an undecided state. Keeping transactions in an undecided state is not a good idea, since any money that is locked in such transactions can not be used by anyone: it leads to a financial loss for all involved parties, and a decrease of capacity for the Ripple network. The only alternative is: it must be possible to roll back a transaction in all its stages, until the final commit decision. There must be a final commit decision after which no rollback is possible anymore: otherwise, the participants of a transaction will never have certainty over the fractions of the shared accounts that are assigned to them, and the fractions they can safely assign to their neighbors.

2.4.2 Atomic vs. non-atomic commits

Some Ripple concepts use a non-atomic commit scheme: the transaction is committed piece-by-piece, one connection at a time. However, if this process takes too long, the not-yet-committed part of the chain can decide to roll back, so the transaction ends up in a partially-committed, partially rolled back state. On the boundaries between the "committed nodes" and the "rolled back" nodes, there are nodes that benefit from this, and nodes that lose from this. Typically, committing is done in the from-payee-to-payer direction, so that the node that doesn't continue the commit will be the losing party, but even then it is possible for malicious parties to set up transactions where they will be the benefiting

side, and some other (possibly unknown) party will end up as the losing side[2]. Because of this, non-atomic commits are a bad idea.

Commits in a Ripple network can be made atomic by having a transaction document that has to be signed by all participants of the transaction in order to commit the transaction. The transaction document can specify several things, e.g. the amount to be transferred, and the public keys of all participants. Each neighbor pair will only consider the transaction to be committed for their connection if they have the transaction document together with all its signatures. Effectively, the moment the last participant signs the document is the moment when the transaction is committed.

A remaining problem in this concept is the transmission of signatures between the participants, and especially the last signature, which determines whether the transaction has been committed. Typically, they will be transmitted in the payee-to-payer direction: this is the direction where it is in the interest of nodes to make sure the next node decides to commit. In the end, however, the same problem remains as in non-atomic commits: if transmission of the last signature is too slow (or starts too late!), a part of the chain decides to commit while another part decides to roll back, and the situation can be exploited in basically the same way as non-atomic commits.

2.4.3 Using Bitcoin

The fundamental problem is how to reach consensus about the commit decision in a decentralized way. It is essential to realize that this has already been solved in Bitcoin, although the solution is a bit slower than desired for e.g. POS systems. Luckily, the Bitcoin solution can be used in a way that solves problematic transactions, while non-problematic transactions are not slowed down by it.

The idea is that, when a participant of the Ripple transaction didn't receive all signatures, and some time-out has passed so he wishes to roll back the Ripple transaction, he creates a Bitcoin transaction with an output that can be spent in the following ways:

- by providing all signatures of the Ripple transaction, or
- by providing a signature, which can only be provided by the participant who created the Bitcoin transaction

All participants keep monitoring the Bitcoin block chain, and look for all occurrences of this type of transaction that correspond to the Ripple transaction they are participating in. If a participant has all signatures of the Ripple transaction, he will use them to spend the output of the Bitcoin transaction as fast as possible. Typically, there will be multiple participants who have all signatures, so they will all try to spend the output of the Bitcoin transaction. One

[2]This is done by arranging a transaction where the malicious node is payee (and optionally also payer), waiting until the transaction is almost timed out, and then asking the direct neighbor of the payee to commit the transaction.

of these attempts will end up in the Bitcoin block chain. Since this will give all participants access to all signatures, this will commit the transaction.

After monitoring the block chain for a couple of blocks, without seeing its output being spent, the original creator of the Bitcoin transaction decides to spend it, using the second method. As soon as this ends up in the block chain, other participants see it, and use it as proof that the Ripple transaction has been rolled back. This proof will be considered to have higher authority than having all signatures of the transaction.

A non-problematic transaction will be decided very fast: as soon as a participant has received all signatures, he knows that the transaction is committed. However, he will have to continue monitoring the block chain, in case one of the participants attempts to roll back the transaction. Stopping such an attempt is easy if you have all the signatures, but it has to be done; especially the payee has an interest in doing this; he might want to use redundant hardware to make sure he is not temporarily disconnected from the Bitcoin network.

The participants have to agree in advance on the maximum block chain length where rollback attempts are still accepted. As soon as this length is reached without any rollback, participants who have all signatures can permanently consider it to be committed. Participants who don't have all signatures can continue to ask their neighbor for them; preferably the one on the payee side, since that one has an interest in a comitted transaction. If this doesn't reveal all signatures, the situation has to be resolved manually. However, it is expected that this will be an extremely rare situation, since all participants are capable of preventing it.

The participants also have to agree in advance to the minimum number of blocks between the Bitcoin transaction and the moment when it is spent in "rollback" mode. This minimum will give other participants an opportunity to spend it in "commit" mode. Typically, a Ripple transaction will specify the following:

- Maximum block index of "rollback attempt" (typical value: current block chain length + 2)
- Minimum block distance between "rollback attempt" and "rollback" (typical value: 3)
- Maximum block distance between "rollback attempt" and "rollback" (typical value: 5)

The amount of bitcoins in a rollback attempt should be as low as possible, since it provides an incentive for the "last signer" of the Ripple transaction to keep his signature secret for others, so that he is the only one who can claim the bitcoins of a rollback attempt. If possible, this amount should be zero.

If multiple participants decide to create rollback attempts, it is theoretically possible that some will be spent in a way that indicates "commit", and others in a way that indicates "rollback". There has to be some protocol to decide between these. One way would be to say that the one with the lowest transaction hash

value is the one that decides between commit and rollback. The protocol itself is not important, as long as it can be evaluated by all participants, and leads to the same conclusion for all of them.

Chapter 3

Scalability analysis

In this section, several payment concepts will be compared for scalability and the feasibility of global-scale usage. Because of the large number of factors involved, the big-O notation will not be used: instead, the order of magnitude of processing, storage size etc. will be estimated for a single case: use by an entire world population of 10^{10} people.

3.1 Assumptions

3.1.1 Number of transactions

currence.nl[1] shows the following numbers for 2008 in the Netherlands, which is a country where electronic payment has replaced cash to a high degree:

- Number of "PIN" (debit card for POS systems) payments: $1.756 \cdot 10^9$
- Number of "Chipknip" (electronic cash for POS systems) payments: $1.76 \cdot 10^6$
- Number of direct debit payments: $1.226 \cdot 10^9$
- Number of "Acceptgiro" payments: $171.6 \cdot 10^6$ (used in 20% of all online purchases, but used for other purposes as well)
- Number of "iDeal" (internet banking) payments: $28 \cdot 10^6$ (used in 35% of all online purchases)

In a news article[2], which also has Currence as source, it is reported that in 2010 there were $2.1 \cdot 10^9$ "PIN" transactions, and almost $3 \cdot 10^9$ cash transactions.

From these numbers, the following can be estimated for the Netherlands:

- Number of POS transactions per year: $5 \cdot 10^9$

[1] http://www.currence.nl/Downloads/Cu_CN_2009_01.pdf
[2] http://www.nu.nl/economie/2422599/pinnen-gaat-winnen-van-contant.html

- Number of online purchases per year: $80 \cdot 10^6$
- Number of other transactions per year: at least $1 \cdot 10^9$

As expected, the number of online purchases is insignificant w.r.t. the number of POS transactions, but there is still a large number of transactions that doesn't fit in either category. While these statistics mostly count end consumer transactions, the number of business-to-business transactions is probably lower than this (when not taking high-frequency trading into account): most of them will be done manually, so it is someone's job to perform them. So, the number of business-to-business transactions is limited by the number of people performing them, and only a small percentage of the population has a job position that involves making a high number of such transactions.

With the "other transactions per year" category, business-to-business transactions and some cash microtransactions that weren't accounted for in the Currence statistics, it doesn't sound unreasonable to estimate the total number of transactions as 10^{10} per year. For $16.5 \cdot 10^6$ citizens, this is 1.7 transaction per person per day, which doesn't sound unreasonable either. Rounding upward to 2 transactions p.p.p.d., this gives a total number of $20 \cdot 10^9$ transactions per day for a future highly developed world population of 10^{10} people.

3.1.2 Number of "unspent transactions"

For the scalability of Bitcoin, it is important to know how many "unspent transactions" there are: all unspent transactions need to be stored, and this takes more storage for lots of small transactions than for a few large transactions. For an estimation, let's use some US economic data:

- Population: $300 \cdot 10^6$
- Money supply (M2, 2011)[3]: $9 \cdot 10^{12}$ dollars

So, that is 30000 dollars of money supply per person. Since most transactions are small, we can estimate the total number of "unspent transactions" by estimating the size of the smallest transactions; in 2011, this is probably around 1 dollar. So, if the US economy ran on Bitcoins, there would be approximately 30000 unspent transactions per person. For a worldwide economy of $10 \cdot 10^9$ people, that would be $300 \cdot 10^{12}$ unspent transactions.

On the other hand, such small transactions are typically collected relatively quickly (e.g. daily or weekly), and most long-term savings are probably filled with larger transactions, e.g. monthly salary. When dividing the money supply by a montly salary of 2000 dollars, we get 15 unspent transactions per person, or $150 \cdot 10^9$ unspent transactions worldwide.

The second estimate is so much smaller than the first estimate that, even if only a small fraction of the small transactions stays unspent for a long time, it will still dominate the total number of unspent transactions. Because of this, the number of unspent transactions is estimated to be 10^{13}.

[3]Source: https://en.wikipedia.org/wiki/File:MB,_M1_and_M2_aggregates_from_1981_to_2012.png

3.1.3 Degrees of separation

For the length of paths in Ripple networks, it is assumed that the "six degrees of separation" law is true. The actual path length might be shorter if some amount of centralization is present, or longer if path-finding does not work efficiently.

3.2 Centralized payment networks

Electronic POS transactions are typically performed through a very small number of payment processors, such as VISA and MasterCard. Transaction data can be sent more or less directly between POS terminals and the server farms of the payment processor.

The bank network mostly follows a tree structure: individual banks are connected through national wire transfer networks, which are connected through the international SWIFT network. One could argue that a payment processor network, which has a star shape, is a special case of the tree shape: a payment processor network only has two levels (customer and payment processor) instead of the multiple levels of the international wire transfer network. In both cases, the number of levels in the tree is very low: the maximum is approximately four.

For every transaction, the following has to be done:

- On every level of the tree: (cryptographic) authentication of the transaction
- On every level of the tree: check whether account balances are sufficient
- On every level of the tree: adjustment of account balances
- When desired: storage of transaction information (assumption: stored for 1 year)

In the multi-level wire transfer network, inter-bank transactions are usually bundled together into large "bulk transactions" between banks, which can happen e.g. once per day. This is one of the reasons why wire transfers take such a long time. This bundling has the effect of nearly eliminating the need for processing or storage on higher levels of the tree.

Processing per person per day	Cryptographic operations	10^0
	Balance look-up and update	10^0
Communication per person per day	Network transfer of transaction data	10^0
Storage per person	Account balance	10^0
	Transaction data	10^3

3.3 "Full" Bitcoin

Here, the possibility is considered of everybody using a "full" Bitcoin node. Every node stores all block headers and verifies every new block, including all

new transactions. It is still assumed that fully spent transactions are "pruned". Without this pruning, transaction storage would grow continuously, without ever reaching a steady state.

Since everybody verifies and stores everybody else's transactions, the number of checks and the amount of storage space equal the total number of transactions. Since everybody needs to receive every transaction at least once, the amount of network data to be transferred also equals this number (assuming the peer-to-peer network is efficient). The amount of transaction data to be stored per person equals the number of unspent transactions.

For the number of cryptographic operations, the mining "proof-of-work" will be ignored. It is expected that, when the number of Bitcoin users is large, the average "proof-of-work" difficulty *per Bitcoin user* can be very low, without threatening the security of Bitcoin. Storage space of the block headers is also ignored.

Processing per person per day	Cryptographic operations	10^{10}
	Transaction balance check	10^{10}
Communication per person per day	Network transfer of transaction data	10^{10}
Storage per person	Account balance	0
	Transaction data	10^{13}

3.4 "Two-tier" Bitcoin

It is possible to have a future Bitcoin system where block storage and verification is only done by a small "elite" of miners and service providers. It is not clear yet to what extent the "elite" would have an opportunity to abuse its position to excercise power over the rest of the population. For simplicity, it is assumed here that having 10^3 independent members in the "elite" provides sufficient competition to protect the rest of the population against abuse.

Processing per person per day	Cryptographic operations	10^3
	Transaction balance check	10^3
Communication per person per day	Network transfer of transaction data	10^3
Storage per person	Account balance	0
	Transaction data	10^5

3.5 The Ripple, without using Bitcoin

With the "six degrees of separation" assumption, there are typically seven participants in a transaction. For storage of transaction data, the same assumption is made as for centralized payment networks.

Processing per person per day	Cryptographic operations	10^1
	Transaction balance check	10^1
Communication per person per day	Network transfer of transaction data	10^1
Storage per person	Account balance	10^0
	Transaction data	10^3

3.6 The Ripple, with using Bitcoin

For evaluating the suggested combination of the Ripple and Bitcoin, it is important to estimate the fraction of transactions that will end up in the block chain. It is expected that, as soon as people have a more or less stable financial situation, their Ripple links almost never have to change (less than once per person per year). Deliberate transaction failure will also be rare, if there is no incentive for creating such failures. It is expected that accidental failures will be the most common, mostly due to communication failure. Since failure has to occur somewhere *during* a transaction, even this will be rare. The number of in-blockchain transactions will be estimated as one per person per year, or a total of $27 \cdot 10^6$ per day.

The number of unspent transactions will also be much lower than in a normal Bitcoin system, since most transactions will take place outside the block chain. However, since every Ripple link corresponds with an unspent transaction, it will not drop as much as the number of in-blockchain transactions. Every person will probably add some unspent transactions to the system.

Processing per person per day	Cryptographic operations	10^7
	Transaction balance check	10^7
Communication per person per day	Network transfer of transaction data	10^7
Storage per person	Account balance	10^0
	Transaction data	10^{10}

3.7 Conclusions

While the proposed system is definitely not the most efficient payment system for worldwide usage, it is a lot better than the "full" Bitcoin concept. In fact, the numbers suggest that the proposed system is already feasible with present-day (2013) personal computer hardware, while this is not the case with the "full" Bitcoin concept (at least not at this scale). While possible on present-day personal computers, it would still take quite a large percentage of system resources (especially storage), making the use unattractive. This does not have to be a real-world problem:

- Bitcoin is still far away from being used on such a large scale. As long as it is still a small-scale system, resource consumption is much lower - in fact, this is why the "full" Bitcoin concept is currently still possible on personal computer hardware, even without transaction pruning. It will take several years to grow to such a huge scale, and in these years, the cost per amount of resource (e.g. per byte of storage space) is expected to drop a lot.

- There is no reason why the proposed system can not be combined with the "two-tier" Bitcoin concept, causing a further reduction in per-person resource usage. In fact, this is expected to happen, if many people decide to let third parties host their Ripple nodes. The reduction of block chain

size because of the Ripple system means that less centralization is needed to have the same efficiency as a "pure" Bitcoin system.

When comparing all concepts, a trade-off is visible between having a low-trust system on one hand, and having a resource-efficient system on the other hand. Clearly, in the absence of trust, more resources are needed as people continuously need to verify each others' behavior.

Chapter 4

Future work

4.1 Remaining attack methods

The most important remaining issue is how to prevent denial of service attacks. No matter how robust the proposed final rollback mechanism is, the to-be-transferred money is still locked for a considerable amount of time, if one of the participants refuses to give his signature. The problem is worsened by the use of anonymity measures, which also provide anonymity to the attacker. It is unknown whether it is possible to identify malicious nodes, even in the presence of the anonymity measures, and to stop routing transactions through that node. It might help that many nodes will typically use the same host together, and the behavior of a node is determined by the software of the host: a malicious node necessarily has to run on a malicious host. This reduces the problem to identifying and avoiding malicious hosts.

4.2 Transaction fees

This document does not cover transaction fees. However, because of the risks involved with transferring other peoples' transactions (at least the risk of making money inaccessible for some time), it might be necessary to pay transaction fees to intermediate nodes. Some Ripple concepts accumulate all fees over the entire route, and let payee or (usually) payer pay all transaction fees. An alternative is the way how internet access costs are distributed: let direct neighbors agree with each other about transaction fees, but don't distribute (information about) these costs across the network. It is unknown what the full effects are of either choice.

4.3 Interoperability

This document focuses on a Bitcoin-only network. It might be interesting to provide gateways to a more traditional Ripple network that (also) deals with other currencies. To make this possible, either the protocol of the Bitcoin-only network has to incorporate all traditional Ripple features, such as currency conversion, or users of the Bitcoin-only network need to make use of special gateway service providers to have access to the traditional Ripple network. In the second case, a transaction that goes through both networks simply consists of two sub-transactions, one for each network.

In actual software implementations, it is probably possible to delay the difficulty of dealing with interoperability, by letting the software support multiple "transaction protocols", and have a sort of plug-in architecture for adding other transaction protocols. This would allow all Ripple variations to join into one huge, technologically heterogeneous network. The advantages and disadvantages of this approach need to be listed.

4.4 Forked transaction paths

Throughout this document, it has been assumed to some degree that a transaction is performed through a linear chain of participants. For transferring larger amounts, it might be useful to allow transactions to *fork*: allow intermediate nodes to split up the total amount into smaller pieces, and let each piece travel through a different route. It is unknown how difficult it is to implement this and how much benefit it will actually provide.