Network Partitioning Effects on Ripple Transactions

Yoan Martin

School of Computer and Communication Sciences

Decentralized and Distributed Systems lab

Semester Project

May 2019

Responsible
Prof. Bryan Ford
EPFL / DEDIS

Supervisor
Cristina Basescu
EPFL / DEDIS

Supervisor
Kelong Cong
EPFL / DEDIS

Contents

1 Introduction

Blockchain is a hot topic. It is used for many activities, but sending money is a significant one. Ripple is a blockchain based network that allows anyone to transfer money across the globe. Since this network is widely used by banks and financial institutions [1], it becomes attractive to attackers. This project considers network attacks executed by a malicious Autonomous System (AS). The latter is a subnetwork of the internet managed by one or more operators [2]. The goal is to understand if an attack on the internet network can affect the behaviour of the Ripple network. To analyse this effect, the Ripple network will be recovered and matched with existing IP data. Then, using a dataset of transactions, the behaviour of the Ripple network can be simulated and analysed. Consequently, this project is a combination of data-analysis and security. Having a good understanding of data-analysis tools and network security is a main requirement to understand and lead this project. This report will explain the implementation of this project. I will start by defining what is the Ripple network and how it is working. Then, the implementation of the graph recovery will be explained. I describe the main strategy and the limitations encountered. Finally, I will analyse this graph by considering different network attacks.

2 What is Ripple?

Ripple is a global payments network. It allows its users to send money in any currency, like USD or CHF. However, Ripple also has its own currency called XRP. The latter is a blockchain-based currency which uses the XRP ledger [3].

2.1 Gateways

As in the real life, any user does not have a direct access to the payment network. He has to use an entry point like a bank or any financial institution. In Ripple, this entry point is called a gateway or an issuer.

2.2 Transaction

When a user A sends money to a user B, the transaction is done in two steps. First, the transaction must be authorised by the network. It means that A sends its transaction intention to the network via its gateway. Then, the Ripple network uses a pathfinding algorithm to reach B. When the path is found, the transaction is validated by the validation network and added to the XRP ledger. Second, the transaction is executed by the network [4]. Note that the XRP ledger contain transactions for all the currencies and not

only XRP. If A and B do not use the same currency, the conversion is done using a book offer [5]. The latter is the equivalent of a currency exchange.

2.3 Validation

A transaction is validated by the consensus protocol. The latter is executed by the validators. They observe the XRP transactions and discuss with other validators to add them to the ledger. The consensus runs in two steps. First, each validator chooses a valid set of transactions to be added to the ledger. Second, all the validator proposals are considered. The transactions with a majority of acceptance are added to the ledger[6]. This consensus protocol behaves correctly if at least 80% of the validators are not malicious [7].

3 Design & Implementation

The main strategy of this project has been inspired by [8]. This previous work analysed the effect of routing attack on Bitcoin. The aim of this project is to do the same kind of analysis for Ripple. This is done by building the internet subnetwork containing all the Ripple servers and intermediaries. This subnetwork considers only the ASes and not all the routers. It is done by matching two datasets: one containing the Ripple servers and one containing the entire internet. When the match is finished, the result is a graph representing the Ripple network from an AS point of view. The next step is to simulate the network by replaying a set of transactions when a malicious AS attacks the network.

3.1 Build the Ripple graph

3.1.1 Recover the Ripple dataset

Ripple provides an API to access public data. This API has been used to extract gateways information. The function in [9] was used to recover the name, account number and IP addresses of each gateway. Then, the IP addresses are used to extract the AS of each gateway. This information is recovered by using the RouteViews [10] dataset. The latter observes the BGP announcements to extract the AS of an IP prefix. Hence, by finding the longest prefix match of a gateway IP address, the corresponding AS is recovered. The final result contains a relation between a gateway account and an AS number. This implementation can be found in [11].

3.1.2 Recover Caida dataset

Recovering the ASes of the gateways is not enough to build the Ripple graph. Since these ASes are not directly connected, the intermediary ASes must be added to the graph. To do so, the entire internet network must be

considered. This information is already created by an organisation called Caida. The latter has created a dataset [12] containing all the ASes and their corresponding links. A link is a physical connection between two ASes with a relationship type. This type can be peer-to-peer, customer-provider or provider-customer depending on the business contract between these ASes. This dataset is well-formatted and directly used as a pandas dataframe. This implementation can be found in [13]

3.1.3 Matching Ripple and Caida datasets

Now that the Ripple and Caida dataframes are recovered. They can be matched to build the final Ripple graph, e.g. the graph containing the Ripple gateways and intermediaries. The latter is build using the NetworkX library [14]. It allows to create a graph from nodes and edges and it provides algorithms to efficiently analyse this graph.

Actually, two graphs are built: The Caida graph C and the Ripple graph R. C is built using the Caida dataset. The ASes connections are used to build C and the relationship is stored as attribute of each edge. R is built in two steps. First, the Ripple dataset is used to add the gateways nodes in the graph. At the end of this step, R contains nodes without any edges. Second, C is used to complete R. Indeed, for each pair of nodes (N_1, N_2) in R. The set of possible paths between N_1 and N_2 are computed in C [15]. Each path is a list of nodes connected by edges. Hence, this list is added to R to build the final Ripple graph. At the end, the graph represents a portion of the internet containing the Ripple ASes, their intermediaries and their connections.

3.2 Process provided transactions

Now that the final Ripple graph is recovered, a tool is needed to simulate and analyse its behaviour. In this project, this tool is a set of transactions provided by the lab. It contains a history of Ripple transactions between 2013 and 2017. The information describing sender, receiver, amount, currency and date are kept for the processing.

Process senders and receivers The senders and receivers are Ripple accounts. Since the Ripple graph is based on gateways, only the transactions with gateways as sender and receiver are kept, see section 4.2. Then, since the gateways nodes in the recovered graph are defined by an AS, senders and receivers accounts are replaced by their corresponding ASes.

Process amount and currencies The transactions are described by many currencies, they should be converted in a single currency to facilitate

the analysis. Since the most frequent currency is XRP, all the transactions are converted in this currency. To do this, an exchange table is built using the Ripple API function in [16]. By providing the date, the origin and the destination currency, the function returns an average exchange rate. Finally, by using this table, all the transactions are converted in XRP. This implementation can be found in [17].

3.3 Replay transactions

Now that the Ripple graph is recovered and the transactions are processed, the behaviour of the Ripple network can be simulated. Actually, each transaction is defined by a sender AS and a receiver AS. This tuple is used to extract a path in the Ripple graph. This path is recovered in two steps. First, a set of shortest paths is recovered using NetworkX [18]. Second, the best path is selected using the relationships between two nodes. In this project, a node prefers to send the traffic to its customer, then to its peer and finally to its provider. This allows to extract the best path for each tuple (sender,receiver) in the transactions dataset. Finally, knowing which path is taken by each transaction allows me to simulate the behaviour of the Ripple network. In addition, since the network is constantly evolving. A different graph is built for each month and the transactions set is also split by month. Replaying the transactions actually means replaying the correct transactions with the corresponding graph. Then, the results are aggregated to provide a complete analysis.

4 Problem encountered & Limitations

4.1 Finding IP data in Ripple API

As explained above, only the gateways are considered as the basis of the Ripple graph. Since there are only 15 gateways, this basis is small. The original idea was to include all the Ripple accounts in the graph. Unfortunately, this is not possible due to the lack of information provided by the Ripple API. In fact, the latter provides plenty of data regarding accounts but nothing related to IP. This disables the capability to know which accounts belong to a particular AS.

There are two cases where IP information are provided by the Ripple API: validators and gateways. As explained above, the validator network contains servers which validate some transactions. However, they are not on the path of transactions, they just observe them. Consequently, the validators are not considered in the analysis. Finally, the transactions are assumed to go through gateways. Hence, using only the gateways provides relevant and enough data to replay transactions.

4.2 Finding gateways accounts in the transactions

As explained above, a transaction can be replayed in the final graph only if the sender and the receiver accounts are gateways. This significantly reduces the set of transactions. Actually, they are represented by two different formats. Transactions between 2013 and 2016, T_1, are described by the sender and receiver accounts. Transactions of 2017, T_2, have the same description with information regarding the sender issuer and receiver issuer. Since a transaction goes from a user account to another user account, T_1 does not contain transactions with a gateway as a sender and as a receiver. Regarding T_2, data using issuers are useful in this project. Since, an issuer is a gateway, T_2 can be used to extract valid transactions. Unfortunately, the issuers in T_2 are mostly not present in the gateways list. This result is surprising because all the transactions should go through them. My opinion on this is that the Ripple API does not provide enough information. This API provides only one account per gateway. However, I think that gateways have many servers and consequently many Ripple accounts. So, I imagine that the invalid transactions go through gateways, but they are not referenced by the function in [9]. This reduces the amount of transaction from 2'000'000 to 30'000.

5 Attacks

Now that we have a graph representing the Ripple network and a set of transactions, the network's behaviour can be analysed. To see the effect of network partitioning, the transactions are replayed by considering a malicious and attacking AS. In this project, we consider two different attacks: an AS which drops the traffic and BGP hijacking. It is realistic to consider these attacks. They happen often in the real world. Indeed, almost 14'000 attacks have been recorded in 2017 [19]. They can have a real impact on cryptocurrencies. For example, on April 2018, a malicious AS managed to reroute the DNS service of Amazon using BGP hijacking. This attack allows the attacker to steal more than 13'000 USD in Ethereum [20].

5.1 Traffic dropped

A malicious AS can disrupt the network by dropping all the packets it receives. This is considered as an attack if the AS drops the packets voluntarily. However, this can be a regular behaviour. Indeed, an AS encountering internal problems produces the same effect. This behaviour is quickly seen by other ASes which update their BGP rules. Consequently, the packets end up to be rerouted through other ASes in the network.

Implementation In this project, the BGP rules update is instantaneous. It means that, as soon as an AS appears down, the packets are directly rerouted through another path, if it exists. This scenario is not realistic. The network takes some time to find out that an AS appears down. However, this scenario is still acceptable in this project because an AS takes at most 3 minutes to understand that a neighbour is down [21]. This amount of time is small compared to the time window of the transactions dataset. The effect of an AS down on the Ripple transactions is computed using two graphs: G_1 is the original graph and G_2 is the original graph with a corrupted node N which appears down. For each transaction, the best path is computed for G_1 and G_2. If these paths are equal, the transaction is completed. If they are different, the transaction is rerouted. If a path cannot be computed for G_2, the transaction is lost. This analysis is repeated for each node N in G_1.

5.2 BGP hijacking

A malicious AS can send wrong BGP rules to attract the traffic of its neighbours. This allows the malicious AS to observe the packets as in a man-in-the-middle attack. Then, it can propagate the traffic normally to avoid any suspicion. Again, this behaviour is not necessarily malicious. This can be the result of badly formed BGP rules.

Implementation Let's consider the original graph G and a corrupted node N. For each transaction, two path lengths are computed. The length of the correct path and the length of the hijacked path. The latter is the length between the source and $N + 1$. The best path is the one with the smallest size. If the correct path is selected, the transaction is completed. If the hijacked path is selected, the transaction is rerouted. This analysis is repeated for each node N in G.

6 Results

6.1 Transactions analysis

Before going into the results of the attacks, it is important to analyse the transactions. This analysis is useful to understand what comes next. The two main results can be seen on figures 1 and 2. They show how the transactions amount is distributed. In this project, the term transaction amount is defined by the transactions value and not by the transactions count. It means that if the set contains only two transactions, one that sends 100 XRP and one that sends 1 XRP, the transaction amount is 101 and not 2. On figure 1, the transactions are not equally distributed in time. Indeed, more than a third of the transactions amount happens in August. On figure 2, the transactions mainly go through two gateways. The latter is the first

important result of the whole analysis. The ASes 13335 and 19551 contribute a lot in the transactions. It means that the Ripple network can be highly affected if one of these ASes is malicious.

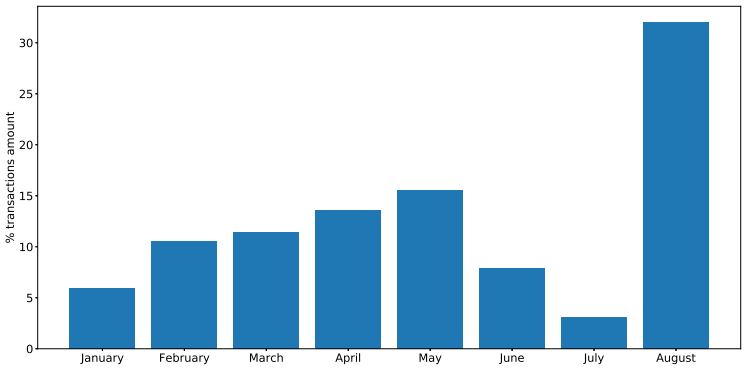

Figure 1: Transactions distribution by month

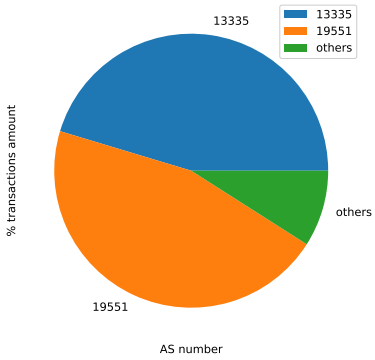

Figure 2: Transactions distribution by gateway

6.2 Analysis when an AS is down

In this analysis, when an AS is down, a transaction can be: still completed, rerouted or lost. Let's start by the case of lost transactions. In the figures 3 and 4, there is a comparison between the transactions lost and the transactions distribution. The figure 3 describes the amount lost by AS in decreasing order. The figure 4 describes the transactions distribution by AS. It is another way to present the result in figure 2. Two important results can

be seen. First, the amount lost is the same as the transactions distribution. This result is not surprising because if the sender or receiver AS is down, the transaction can necessarily not be completed or rerouted. Second, a transaction is lost only if an AS containing a gateway is down. This means that a transaction cannot be lost if an intermediary AS is down. In other words, it means that an alternative path can always be found.

Let's follow by the case of rerouted transactions. As it can be seen in figure 5, the rerouted amount is generally small. The only exception stands for the AS 553. It means that this AS is close to the most important gateways. Consequently, if it is down, the transactions need to be rerouted.

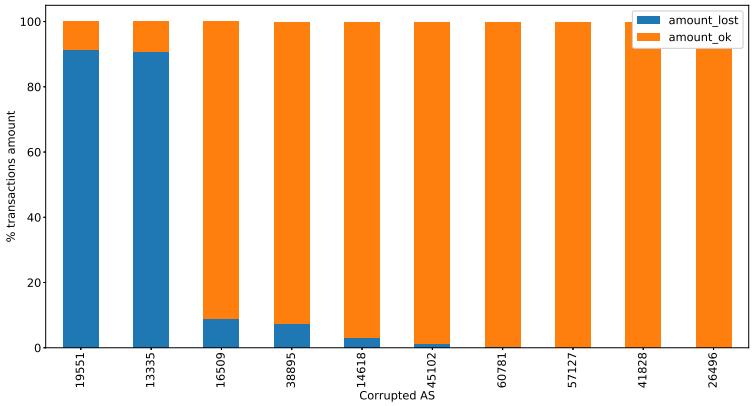

Figure 3: Amount lost when an AS is down

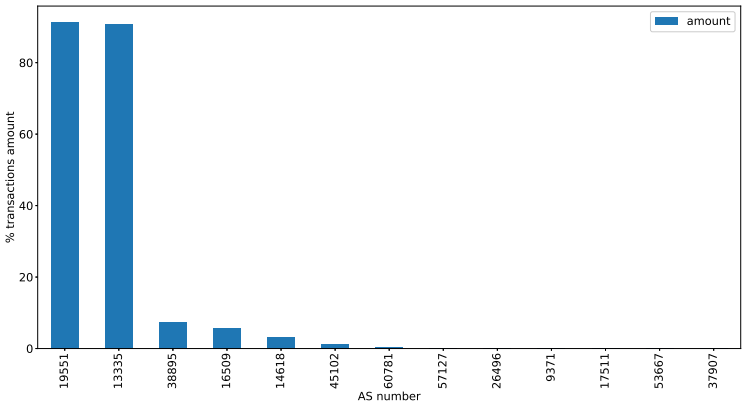

Figure 4: Transactions distribution by AS

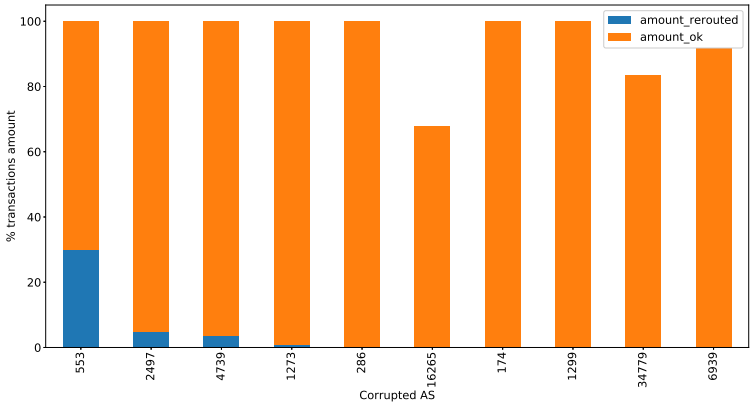

Figure 5: Amount rerouted when an AS is down

6.3 Analysis of BGP hijacking

When there is BGP hijacking, a transaction can never be lost. Indeed, a malicious AS never drops the traffic and the transaction can still find a path. The figure 6 shows that this attack is more important than the first one. As it can be seen, between 30% and 40% of the traffic can be rerouted by a long list of ASes. It means that the attack can have an effect on the Ripple network for many malicious ASes. Note that in figure 5 and 6, some columns look incomplete, e.g. the total amount does not go to 100%. This result is normal. It shows that the corresponding AS only appears in the Ripple graph for some months but not for others. Consequently, when the results are normalised, the column cannot reach 100%.

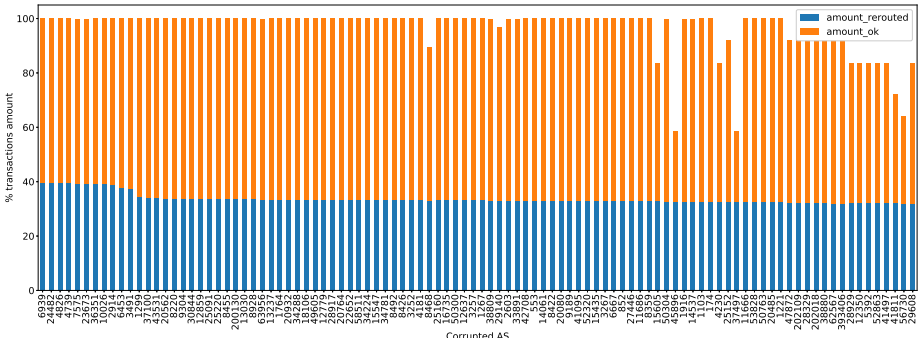

Figure 6: Amount rerouted when BGP hijacking

6.4 Evolution in time

The previous analysis only considers the entire set of transactions. The latter are replayed by considering each AS as malicious. This allows to see which AS can affect the Ripple network. However, this analysis does not say if this effect is important and how it evolves in time. Here, the idea is to see if a month is more affected by an attack than others. To do this, the analysis of each month is aggregated. It means that each month gets three corresponding values: the amount of completed, rerouted and lost transaction. The final result is a dataframe describing the time evolution by month.

AS down The result, when an AS is down, can be seen in the figure 7. As we can see, the amount of lost and rerouted transaction is stable. In details, the amount of lost transaction varies between 4.5% and 5.5% and the amount of rerouted transactions varies between 0% and 2.5%. In addition, these amounts are small. In particular, the last columns shows the result for the whole transactions dataset. This shows that the effect of an AS down remains relatively small on the Ripple network.

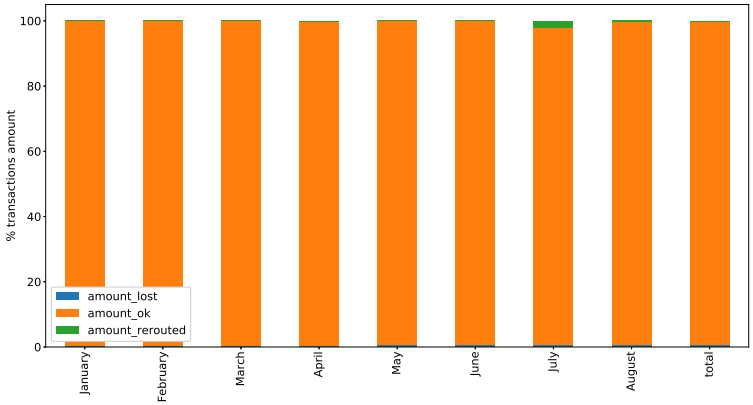

Figure 7: Time analysis: AS down

BGP hijacking The result when there is BGP hijacking can be seen in the figure 8. As it can be seen, the conclusions of the first attack cannot be repeated here. First, the variations of the amount of rerouted transaction are important. Second, the percentage of rerouted transactions can be large. Let's focus on the results of the last two columns.

If the August month is considered, it shows that more than 60% of the transactions are rerouted. This can be explained by the following graph

observation. As seen in the transactions analysis above, an important part of the transactions go through two ASes. The detailed study of the graphs shows that there exists a direct link between these two ASes except for the August graph. Since an intermediary AS cannot provide a better alternative path, it cannot hijack the traffic between two directly connected ASes.

If the total column is considered, it can be seen that BGP hijacking affects fairly little the Ripple network. However, this attack is more important than the case of an AS dropping the traffic.

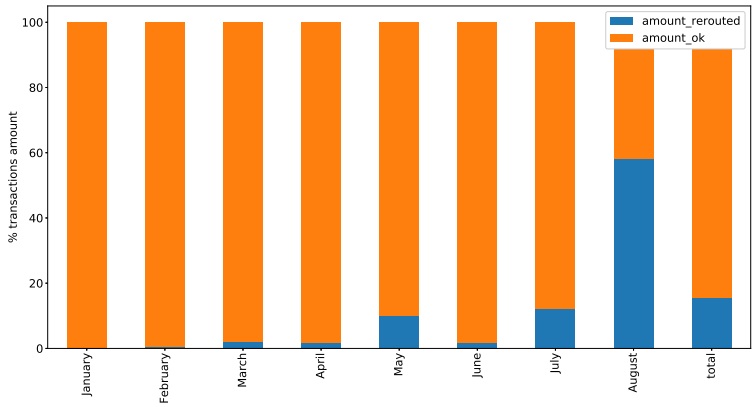

Figure 8: Time analysis: BGP Hijacking

7 Conclusion

As a conclusion, it has been seen that extract IP information from Ripple is a difficult task. In fact, the only usable information is the IP addresses of the gateways. This allows to extract the AS number in which each gateway is located. Then, the Ripple network can be matched with the internet network provided by Caida. When the internet subnetwork used by Ripple is isolated, the behaviour of this subnetwork can be simulated by replaying a set of transactions. So, the effect of a malicious AS can be introduced in this simulation. In this project, two attacks have been considered: an AS dropping the traffic and BGP hijacking. The analysis of these attacks shows that their general effect is relatively small. Nevertheless, it also shows that there exists an important point of failure. Since most of the traffic go through two ASes, the effect can be much bigger. First, if one of them becomes malicious and drops the traffic, an important part of the transactions are lost. Second, if the link between these two ASes is broken, it allows an intermediary AS to reroute the traffic using BGP hijacking. However, since

the transactions dataset cannot have not been used entirely, it can affect the results presented in this project. Consequently, they should be verified using another set of transactions.

8 Future work

8.1 Explore validation network

As we have seen in the definition of Ripple, the transactions are validated by the validation network. In this project, this network has not been considered. However, this network behaves correctly if at least 80% of the servers are not malicious. It could be interested to analyse the effect of a malicious AS on this validation network. If an attack achieves to compromise the validation process, the Ripple network can be highly affected.

8.2 Extract more nodes from Ripple

As we have seen, this project is limited by the IP information provided by the Ripple API. However, it could be interesting to investigate this problem in details. An idea could be to create a Ripple account and send some transactions. By observing the packets, it would be possible to see if a gateways has many IP addresses. This would allow to improve the analysis of this project.

9 Code installation

Since this project concerns data analysis, there is no need to install any dependencies and run the code. My job is accessible at `https://github.com/dedis/student_19_ripple-partition`. Each part of the project is split in different notebooks that are already executed. They can be read directly on Github.

References

[1] M. Down, *200+ Banks & Financial Organizations will use RippleNet in 2019*, Jan. 2019. [Online]. Available: https://hackernoon.com/200-banks-financial-organizations-will-use-ripplenet-in-2019-95d20cc6bb94 (visited on 06/04/2019).

[2] *Autonomous System Lookup (AS / ASN / IP)*, en-US. [Online]. Available: https://hackertarget.com/as-ip-lookup/ (visited on 06/05/2019).

[3] *The Difference Between Ripple and XRP*, en-US, Jul. 2018. [Online]. Available: https://ripple.com/insights/difference-ripple-xrp/ (visited on 05/20/2019).

[4] Crypto Currency, *How Ripple Works xCurrent*. [Online]. Available: https://www.youtube.com/watch?v=vwM-BGLsuZQ (visited on 05/20/2019).

[5] *Offers - XRP Ledger Dev Portal*. [Online]. Available: https://developers.ripple.com/offers.html (visited on 06/05/2019).

[6] B. Chase and E. MacBrough, "Analysis of the XRP Ledger Consensus Protocol," en, *arXiv:1802.07242 [cs]*, Feb. 2018, arXiv: 1802.07242. [Online]. Available: http://arxiv.org/abs/1802.07242 (visited on 05/20/2019).

[7] *Introduction to Consensus - XRP Ledger Dev Portal*. [Online]. Available: https://developers.ripple.com/intro-to-consensus.html (visited on 05/20/2019).

[8] M. Apostolaki, G. Marti, J. Mller, and L. Vanbever, "SABRE: Protecting Bitcoin against Routing Attacks," en, *arXiv:1808.06254 [cs]*, Aug. 2018, arXiv: 1808.06254. [Online]. Available: http://arxiv.org/abs/1808.06254 (visited on 06/05/2019).

[9] *Ripple Data API v2 - XRP Ledger Dev Portal*. [Online]. Available: https://developers.ripple.com/data-api.html#get-all-gateways (visited on 06/04/2019).

[10] *Routeviews University of Oregon Route Views Project*. [Online]. Available: http://www.routeviews.org/routeviews/ (visited on 06/04/2019).

[11] *Contribute to dedis/student_19_ripple-partition development by creating an account on GitHub*, original-date: 2019-02-20T11:29:18Z, Jun. 2019. [Online]. Available: https://github.com/dedis/student_19_ripple-partition/blob/master/Ripple/Gateway.ipynb (visited on 06/07/2019).

[12] C. C.f.A.I. D. Analysis, *AS Relationships*. [Online]. Available: http://www.caida.org/data/as-relationships/index.xml (visited on 06/04/2019).

[13] *Contribute to dedis/student_19_ripple-partition development by creating an account on GitHub*, original-date: 2019-02-20T11:29:18Z, Jun. 2019. [Online]. Available: https://github.com/dedis/student_19_ripple-partition/blob/master/Caida/AS-Exploration.ipynb (visited on 06/07/2019).

[14] *Overview of NetworkX NetworkX 2.3 documentation*. [Online]. Available: https://networkx.github.io/documentation/stable/ (visited on 06/04/2019).

[15] *Contribute to dedis/student_19_ripple-partition development by creating an account on GitHub*, original-date: 2019-02-20T11:29:18Z, Jun. 2019. [Online]. Available: https://github.com/dedis/student_19_ripple-partition/blob/master/analysis_helper.py (visited on 06/07/2019).

[16] *Ripple Data API v2 - XRP Ledger Dev Portal*. [Online]. Available: https://developers.ripple.com/data-api.html#get-exchange-rates (visited on 06/04/2019).

[17] *Contribute to dedis/student_19_ripple-partition development by creating an account on GitHub*, original-date: 2019-02-20T11:29:18Z, Jun. 2019. [Online]. Available: https://github.com/dedis/student_19_ripple-partition/blob/master/Ripple/Exchange.ipynb (visited on 06/07/2019).

[18] *Networkx.algorithms.shortest_paths.generic.all_shortest_paths NetworkX 2.3 documentation*. [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.shortest_paths.generic.all_shortest_paths.html#networkx.algorithms.shortest_paths.generic.all_shortest_paths (visited on 06/04/2019).

[19] *14,000 Incidents: A 2017 Routing Security Year in Review*, en-US, Jan. 2018. [Online]. Available: https://www.internetsociety.org/blog/2018/01/14000-incidents-2017-routing-security-year-review/ (visited on 06/05/2019).

[20] R. Brandom, *Hackers emptied Ethereum wallets by breaking the basic infrastructure of the internet*, Apr. 2018. [Online]. Available: https://www.theverge.com/2018/4/24/17275982/myetherwallet-hack-bgp-dns-hijacking-stolen-ethereum (visited on 06/05/2019).

[21] S. Hares, Y. Rekhter, and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, en. [Online]. Available: https://tools.ietf.org/html/rfc4271#section-10 (visited on 06/05/2019).