RIM Crypto

Official Technical Data Sheet

V1.0

05 th of June 2018

www.rimcrypto.eu

info@rimcrypto.eu

Table of Content

1. Blockchains

Blockchains as payment systems are basically distributed databases solving the double spending problem. The double spending problem is basically a special case of the byzantine generals problem, which is solved in bitcoins using Proof-of-Work, meaning conflicts are highly unlikely in bitcoins because bitcoin blocks are hard to calculate, therefore concurrency in transactions is prevented. Implementing a blockchain needs asymmetric cryptographic algorithms (Public key crypto system) and cryptographic one-way hash functions. The former is used as a key to access your account and verify you as an owner, the later is used to verify payments related to previous balance. Payments are verified by the blockchain adding verification signatures and hash sums.

2. Blockchain implementation

We will use a blockchain, which is using ledgers instead of UTXOs and distributed agreements instead of Proof-of-Work to solve the double spend problem. We will setup a master server that delegates the network calculations. Different other servers (or miners) may interact with the master server helping its blockchain calculations. This means after an initial setup the master server does not only do the calculations needed for transactions, but is also used to just verify the calculations of other servers, becoming homogeneous to the other servers in the network. Every server in the blockchain network has an own signature and a trust value propagated through the network, meaning if calculations of a server are correct its trust value is increased, if the calculations are incorrect its trust value is reduced. The trust on blockchain transactions is increased by the trust of the servers signing. A specific value of trust is needed for a transaction to be verified. More than half of the summed trust value of all responsive servers is needed for a transaction to be verified. This means the trust value needed for a transaction to be verified will increase with growth of the network. So in the beginning phase for a transaction to be verified it is sufficient that the master server has signed it, because other servers are not trusted in the beginning. Later on the trust value needed to sign a transaction will be much bigger than the maximum trust value of single servers, meaning multiple highly trusted servers have to sign a transaction to have it verified. A transaction can be a money transfer with a structure like this:

```
typedef char key;
typedef char sig;

struct _transfer{
 key sender[KEY_LENGTH];      //senders public key
 key receiver[KEY_LENGTH];    //receivers public key
 unsigned long long amount;   //amount to be tranfered
 sig signature[SIG_LENGTH];   //signature to verify the transaction
};
```

For signing a transaction the server needs to check that the transaction is valid, here is some pseudo C code for such a validity check:

```
int verify_transfer(struct _transfer trans){
 sig* sig_hash=hash(trans);
 sig* sig_decrypt=decrypt(trans.signature,trans.sender);
 if(memcmp(sig_hash,sig_decrypt,SIG_LENGTH)!=0){
  printf("Invalid transfer received: Wrong signature!\n");
  return 1;
 }
 P(change_blockchain); //semaphore for preventing double spending of coins
 if(get_amount(trans.sender)<trans.amount){
  printf("Invalid transfer received: Not enough coins to transfer!\n");
  return 1;
 }
 apply(trans); //modify balance of the accounts on the local blockchain
 sign(trans); //sign the transaction
 V(change_blockchain); //free the semaphore
 return 0;
}
```

Also there is a time limit for transactions to be revoked, so besides checking the transaction is verified by signed trust it should also be waited for a specified time amount to be sure the transaction is really valid. Ripple uses a verification time of 10 seconds, stellar uses a verification time of 5 seconds. We are aiming at similar verification time limits.

Preventing double spending

Suppose an account has 3 coins, and tries to do two transactions spending 2 coins, meaning a total of 4 coins. If a single server receives these transactions only the first transaction is signed, because after the first transaction is signed there are not enough coins left to cover the amount of the second transaction. Now if both transactions were send to different servers both transactions would be signed by two different servers and then propagated over the network competing to first get the value of trust needed to be verified. More than half of the summed trust value of all responsive servers is needed for a transaction to be verified. This means still only one transaction can be verified and the other has to be reverted locally on the servers that signed it. We see it is not possible to double spend coins with this approach of verification of transactions. Now suppose we have malicious servers, then over half of the trust value of the available responsive servers is needed to verify a malicious transaction, this means a malicious attacker would need to run many servers over a long time to gain enough trust to do a malicious attack. But if that happens other trusted servers as the master server can issue a revoke of the transaction, meaning the trust of the malicious transactions is decreased and the time limit that is needed for their verification is increased. The trust value needed for a transaction to be revoked is therefore much smaller than the trust value needed to verify transactions, but if a trusted server revokes a transaction without proper reason its own trust value is decreased below the value needed to to do future revocations. A transaction that is revoked without proper reason (like a concurrent transaction or a revoking signature from the transactions sender) will still get verified after an increased timeout.

Keylength

Current Bitcoin miners have a maximum rate of a few GH/s meaning $k \cdot 10^9$ hashes per second. Now suppose a worst case $k \text{ of } 10^3$ and also suppose we have a keylength of 512 bits, then finding a specific key with the given hashrate within a thousand years has a probability of $\frac{10^{15}\cdot 60^2\cdot 24\cdot 365.25}{2^{512}} \approx 2.353673334589236\cdot 10^{-132}$.
This means it is quite impossible to find a specific key, meaning the keylength of 512 is more than enough to secure a cryptographic wallet. Let us go further and study the possibility of a birthday attack, meaning to exploit the birthday paradox as a possible attack vector. This means that we don't try to guess a specific key, but rather search for a random existing key. This of course means that if all 2^{512} possible keys were in use the probability to find a random key is 100% even though the probability to find a specific key is still as low as calculated above. This means the actual probability to find a random key increases by the number of keys in use. Suppose we have 10 billion humans (10^{10}) and each one uses 10^{20} keys, meaning we have 10^{30} keys total, then the probability for a collision is $1 - \frac{2^{512}!}{(2^{512}-10^{30})!\cdot(2^{512})^{(10^{30})}}$ which is quite impossible to calculate but an approximation gives us a probability of still as small as 10^{-94} . Even if we had $1.6\cdot 10^{68}$ hashes the probability of a collision is just about 10^{-18}. . This means even with the birthday paradox it is quite impossible to find any key. To further increase security we will at least use 1024 bits of keylength which is even more secure.

Transition

The transition from WAVES tokens to an independent RIM blockchain can happen with a simple smart contract, where WAVES tokens that are transferred to a specific address defined by us will be frozen or even burned in the WAVES chain and credited by a fixed exchange-rate to a new public-key address in the new RIM blockchain given by the contract.

3. RIM paycard

The RIM paycard works similar to a normal debit card, the only difference is how payments are done at the bank backend. RIM paycards will use an individual blockchain address. On payments the bank calculates that there are enough RIM coins on that blockchain address using the lower corridor wall as an exchange rate. Payment then happens issuing the amount to a specific RIM blockchain address connected to a real bank account from **RIM Crypto** to transfer the actual money from. This means in case of a payment with a RIM paycard we just buy back your token and transfer the money. The lower corridor of the day is used as the exchange rate. We will then try to sell the tokens received to a higher price to return additional tokens received from the money to your RIM paycard excluding a small trading fee. On abnormal market behaviour (current token price leaving the lower corridor) payments with RIM paycard are disabled. Customers may also set an additional upper corridor limit that disables payments with the RIM paycard if exceeded. As soon as the exchange rate is back within the corridors the payments with the RIM paycard are re-enabled.

4. Stabilization mechanism

Introduction

The goal of this chapter is to lay out some approaches used for machine learning in the stabilization mechanism for keeping **RIM Token** exchange rates in a controlled price corridor. One of this approaches is to use Artificial Neural Networks. Deep learning is a revolutionary machine learning technique based on many layers of artificial neural networks (ANN). Deep learning improved pattern recognition over previous approaches. ANNs consist of artificial neurons, where each neuron is basically a linear dot product over an input vector and a weight vector, followed by a non-linear activation function. An ANN is trained by changing the weight vectors. A simple approach of training a feed forward ANN is to use the backpropagation algorithm, which is basically a stochastic gradient descent over the backpropagated error function, with E being the error, η being a learning w_{ij} rate and being the weight value connecting neuron i to neuron j this means the actual w_{ij} change for calculates as $\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$.

Incomplete data

One important thing to take care of is that we have incomplete data. Especially some values are missing because not all crypto currencies existed over the whole time we are tracking. One example is the **RIM Token** itself because we are just having an ICO and no data exists before **RIM Token** older data from other crypto currencies is also useful. There are different ways to handle missing data such as imputation. A promising approach is outlined in [Tre94] were deficient data is used to improve training results in an neural network. A naive way for dealing with missing values is to fill them with mean values. A more precise way for coping with missing values is using prediction, such as regression for the missing values itself. In the case that there was no currency it is best to choose the values in a way that doesn't influence concurrent values, meaning for that training steps the results should be similar to training without that missing values at all. This is due to the fact that data missing before an ICO is systematic and different from data missing at random, or data missing because of an underlying event that is happened unpredictable but has a real influence on the data.

Handling corridors

We want to train on different corridors. This can either be achieved by doing a prediction on the actual exchange rates and volumes and calculate the corridors based on that or it can be achieved by training on the corridors directly. Training on the corridors directly is only feasible if there are discrete corridor steps to be used, then it would be a good idea to train an ordinal classification on the corridors. When continuous corridors are needed it is better to do a regression on the exchange rate, that could also happen with conformal prediction, meaning we would achieve confidence intervals on an exchange rate. conformal prediction would also have the property that it could be done in an online learning setting, meaning the model could improve continuously on the actual data without explicit retraining.

Worst case simulation using similar token data

In the first phase before the ICO we are doing some worst case simulations to verify the feasibility of our approach. For this simulation we are using real data from similar tokens to calculate how long our system can keep up in the worst case. This is worst case because in a real scenario our upper and lower walls directly influence the exchange rate, which is not the case in this first simulation and therefore it is worse than the actual behaviour. This simulation is implemented in python using pandas

Inference on model trained on similar token data

In the second phase we are training an adaptive model of intrinsic exchange rate behaviour to achieve a more realistic behaviour. This model is already an artificial feed forward neural network trained using TensorFlow. For the phase before the ICO the crawled data is split into training data and test data by excluding a few month from training. The model is trained on the training data and the simulation is then an inference of the trained model over the test data. This will allow multiple improvements over the initial neural network, by changing its structure for better predictions.

Inference over hot data

After the ICO the separation into training data and test data is not strictly needed any more because the structure of the neural network is already verified for the specific type of data. The goal of this phase is to actually use a model trained over the whole data set to predict the current exchange rate of **RIM Tokens**. Of course the machine learning algorithms are still improved in parallel to this application.

Improved training data

For further improvements in the accuracy of our model we will later collect not only the exchange rates but also the actual order books of cryptocurrencies important for RIM, likewise the exchange rates and volumes the asks and bids also also read in JSON format from the Web-APIs and then fed to pandas. Later we will also check common blogs and news entries regarding crypto currencies, to collect textual data which can be included in our training model by using word embeddings like word2vec before feeding them to a neural network. The textual data can either be combined with the numeric model by retraining over the whole data set or by training separate neural networks and stacking them.

5. References and following topics

References

[Tre94] Neuneier Tresp Ahmad. Training Neural Networks with Deficient Data. 1994.

Following topics

In the following version of this Data Sheet, we will provide additional information about the application and some technical information concerning our Full Node.
Additionally, we will publish more information about the current state of our project.

Imprint and Contact

RIM Crypto

Technologiepark 8

33100 Paderborn

www.rimcrypto.eu ⬛ 🟥 🟨 Made in Germany

info@rimcrypto.eu

garage33

Freiraum für Gründer und Unternehmer

TEC UP

UNIVERSITÄT PADERBORN

WAVES

IDnow

mtp

PADERBORN