INSTRUCTIONS FOR THE USE OF CHOUCOIN

MICHAEL CHOU

1. EULA

By using or participating in Choucoin you agree to the following terms:

(1) Michael Chou has the right at any time to roll back to Choucoin ownership at a previous time.
(2) You **will not** under any circumstances trade Choucoin for anything of actual monetary value.

2. Introduction

First, it is recommended you watch the video by 3Blue1Brown entitled "Ever wonder how Bitcoin (and other cryptocurrencies) actually work?" [1] that can be found at the following url:

`https://youtu.be/bBC-nXj3Ng4`

This video will give a basic understanding to the underlying machinery behind "proof-of-work" crypto-currencies that are replicated in the attached Magma code. The magma code can be found on the course website:

`https://sites.tufts.edu/michaelchou/teaching/`
`math-150-mathematical-cryptography/`

A free online Magma [2] calculator is available at:

`http://magma.maths.usyd.edu.au/calc/`

In this document we will cover how to post transactions and how to mine blocks. Most importantly, we will set up an agreed upon syntax used in these actions.

3. Obtaining a public key/private key pair

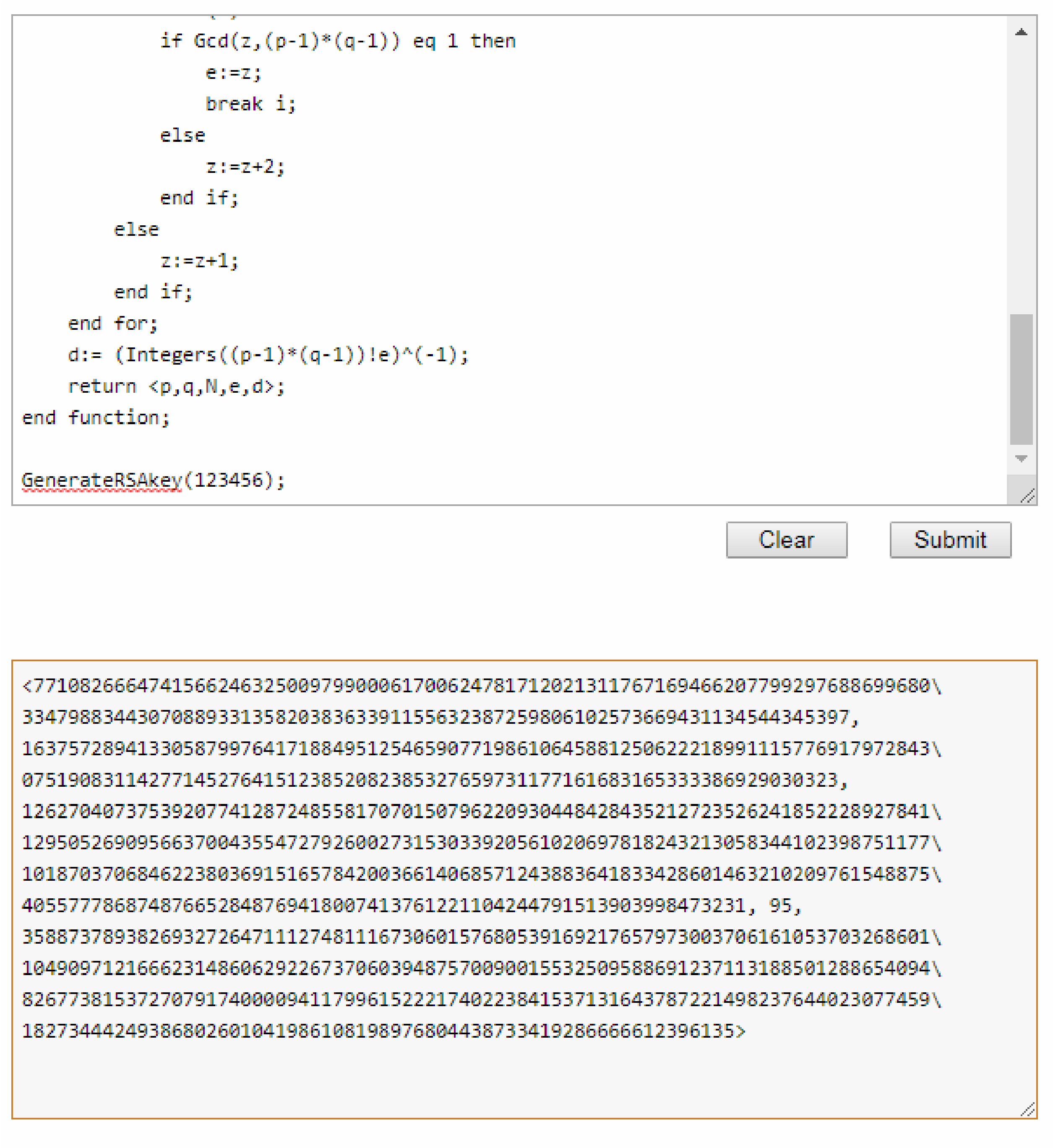
The first step to getting involved with Choucoin is obtaining your own public/private key pair. To do this, you will use the function "GenerateRSAkey()". Copy and paste this function into the Magma calculuator, and then in a new line, simply type

GenerateRSAkey(123456)

where you can choose 123456 to be literally any number you want. I recommend choosing a relatively random number. Magma will return an error if you choose a number greater than 2^32-1. This gives you 5 numbers. Note that the "\" symbol at the end of the line indicates that the number keeps going.

The first number is your first prime p. The second number is your second prime q. The third number is your number $N = p \cdot q$. The next number is e, and the final number is d. In the example

Figure 1. Example of using GenerateRSAkey(123456)



```
            if Gcd(z,(p-1)*(q-1)) eq 1 then
                e:=z;
                break i;
            else
                z:=z+2;
            end if;
        else
            z:=z+1;
        end if;
    end for;
    d:= (Integers((p-1)*(q-1))!e)^(-1);
    return <p,q,N,e,d>;
end function;

GenerateRSAkey(123456);
```

```
<7710826664741566246325009799000617006247817120213117671694662077992976886996800\
334798834430708893313582038363391155632387259806102573669431134544345397,
163757289413305879976417188495125465907719861064588125062221899111577691797284 3\
07519083114277145276415123852082385327659731177161683165333386929030323,
126270407375392077412872485581707015079622093044842843521272352624185222892784 1\
129505269095663700435547279260027315303392056102069781824321305834410239875117 7\
101870370684622380369151657842003661406857124388364183342860146321020976154887 5\
40557778687487665284876941800741376122110424479151390399847323 1, 95,
358873789382693272647111274811167306015768053916921765797300370616105370326860 1\
104909712166623148606292226737060394875700900155325095886912371131885012886540 94\
826773815372707917400009411799615222174022384153713164378722149823764402307745 9\
18273444249386802601041986108198976804438733419286666612396135>
```

given in Figure 1 we have:

$$\begin{aligned} p &= 7710826664741566246\cdots \\ q &= 1637572894133058799\cdots \\ N &= 1262704073753920774\cdots \\ e &= 95 \\ d &= 35887378938269327\cdots \end{aligned}$$

Note that the numbers are separated by commas.

The values $p, q,$ and d are your private key, or secret key (sk). You should under no circumstances share this with anyone else. The values N and e are your public key, and you should share this with the entire class. I will make a chart online that has everyone's chosen public key. Now with these values in hand, you are ready to use the other functions.

4. Posting a transaction

All transactions should be posted at the following message board:

`https://xoyondo.com/mb/vp1FMeH9d3tzW1J`

You may make posts under a name there, but note that anyone can choose any name, so a transaction should not be trusted simply by name alone!

Here is an example of how to give Choucoin (CC) via the message board:

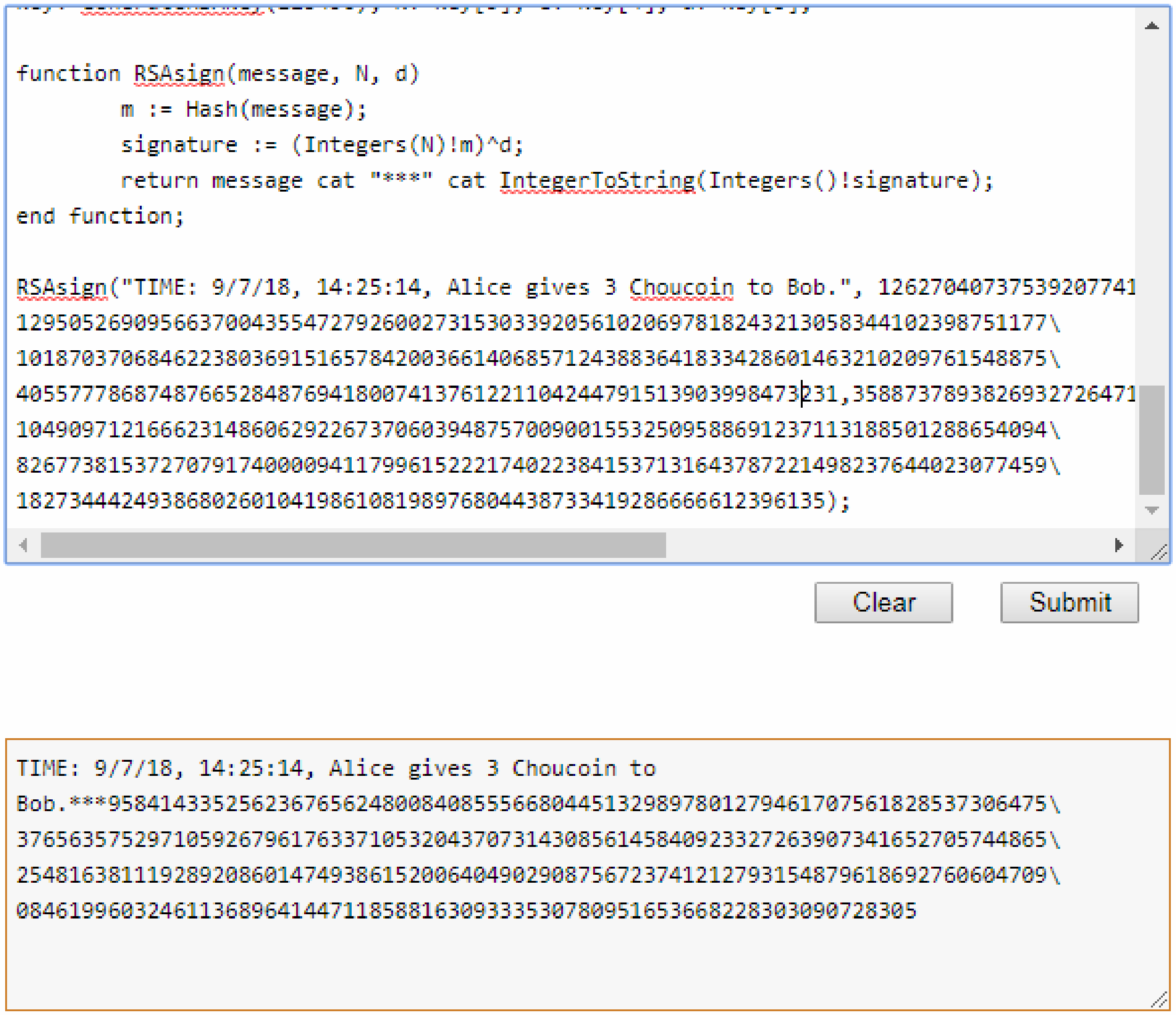
Alice bought an item from Bob for 3 Choucoin. The person who is *giving* the Choucoin needs to go post the transaction. Alice first needs to create the message which will include the time of the transaction. Her message first begins like:

```
TIME: 9/7/18, 14:25:14, Alice gives 3 Choucoin to Bob.
```

However, recall that Alice must digitally sign this message in order for us to believe it is her posting this message. To do this, we use the function "RSAsign()". Copy this function into the calculator and then call the function on your string above, inputing your public key parameter N and your secret key parameter d. Note that the text **must** be in quotations when put into the function. See Figure 2 to continue the example we had from before.

This text output is the actual message we want to post to the transaction forum. Note, when posted to the forum, it should be placed in a single line, and the backslashes used in MAGMA should be removed. **Be careful not to delete any numbers when removing the backslashes.** See Figure 3 to see an example of what it should look like. Formatting is VERY important for the code to work, so do not delete extra spaces or anything like that.

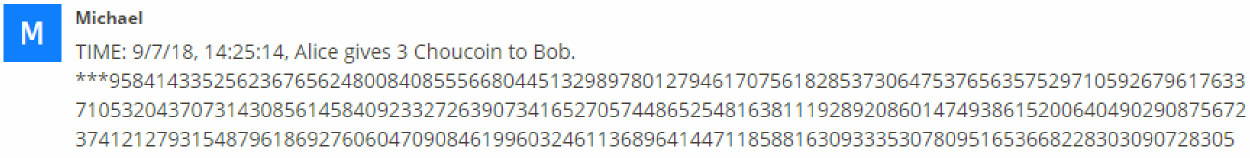
Figure 2. Example of using RSAsign on message



```
function RSAsign(message, N, d)
        m := Hash(message);
        signature := (Integers(N)!m)^d;
        return message cat "***" cat IntegerToString(Integers()!signature);
end function;

RSAsign("TIME: 9/7/18, 14:25:14, Alice gives 3 Choucoin to Bob.", 126270407375392077411
12950526909566370043554727926002731530339205610206978182432130583441023987511777\
10187037068462238036915165784200366140685712438836418334286014632102097615488755\
4055777868748766528487694180074137612211042447915139039984732231,358873789382693272647111
10490971216662314860629226737060394875700900155325095886912371131885012886540944\
82677381537270791740000941179961522217402238415371316437872214982376440230774599\
182734442493868026010419861081989768044387334192866666612396135);
```

```
TIME: 9/7/18, 14:25:14, Alice gives 3 Choucoin to
Bob.***95841433525623676562480084085556680445132989780127946170756182853730647575\
37656357529710592679617633710532043707314308561458409233272639073416527057448655\
25481638111928920860147493861520064049029087567237412127931548796186927606047099\
08461996032461136896414471185881630933353078095165366822830309072830
```

Figure 3. Posting the transaction



Michael

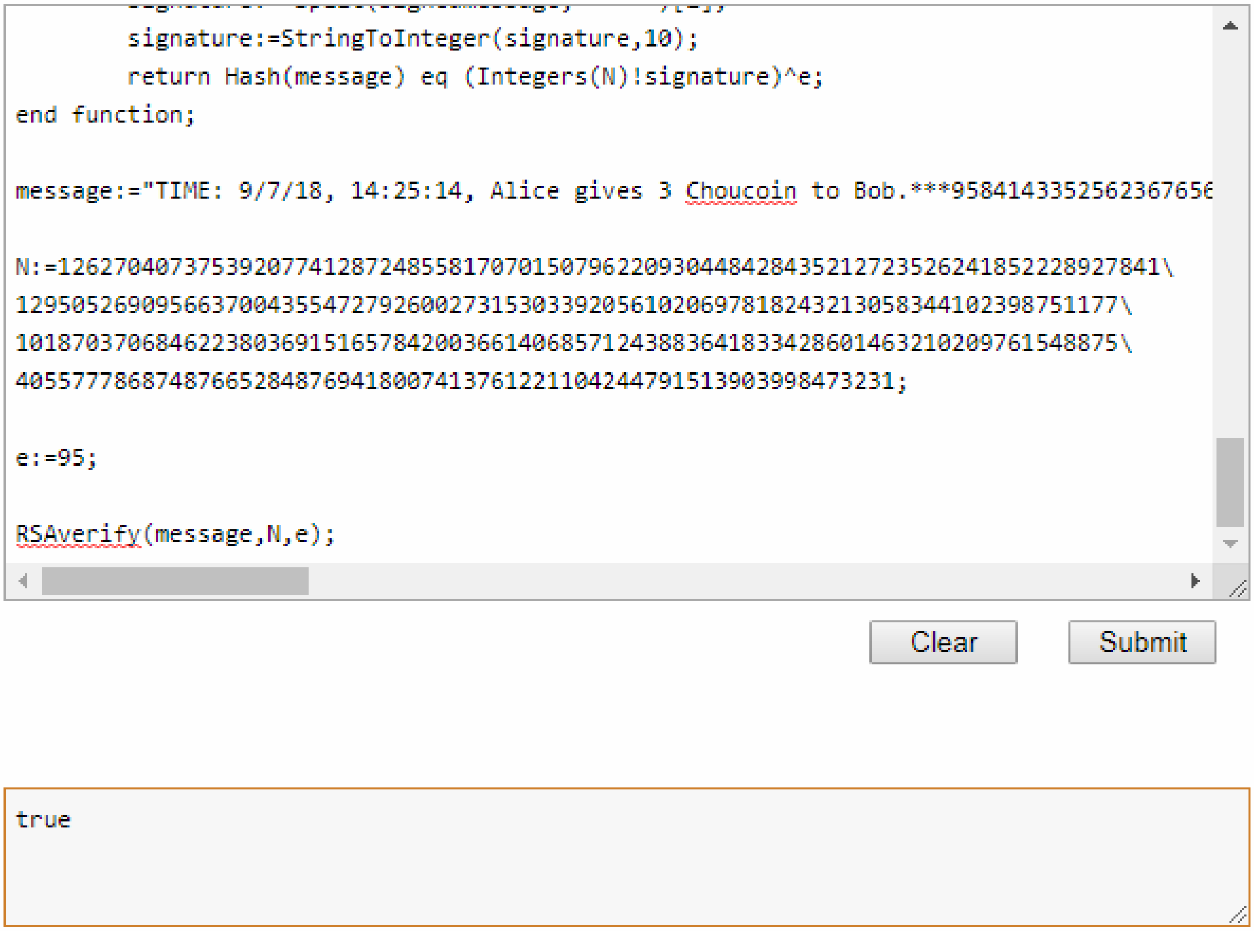
TIME: 9/7/18, 14:25:14, Alice gives 3 Choucoin to Bob.
***95841433525623676562480084085556680445132989780127946170756182853730647537656357529710592679617633
71053204370731430856145840923327263907341652705744865254816381119289208601474938615200640490290875672
374121279315487961869276060470908461996032461136896414471185881630933353078095165366822830309072830

5. Verifying a transaction

Verifying a message is simple, but can be frustrating if the procedure isn't done exactly. First, copy the transaction off the message board. It's simplest to assign the message to a variable in Magma. In Figure 4 we've assigned the message to a variable called "message". Further, we've copied the value of N and e from before, and simply made variables called N and e for them. Finally, copy the function RSAverify into the input, and call the function on the message, N, and e. It will return true if the digital signature is correct, and false otherwise.

CAREFUL: you must make sure that the message is all on a single line. When I copied the message off the message board, an additional line break was introduced. You must delete that line break, otherwise you may get a result of "false" even when it was correctly signed.

Figure 4. Verifying a transaction



```
        signature:=StringToInteger(signature,10);
        return Hash(message) eq (Integers(N)!signature)^e;
end function;

message:="TIME: 9/7/18, 14:25:14, Alice gives 3 Choucoin to Bob.***958414335256237656

N:=126270407375392077412872485581707015079622093044842843521272352624185222892784 1\
1295052690956637004355472792600273153033920561020697818243213058344102398751177\
1018703706846223803691516578420036614068571243883641833428601463210209761548875\
40557778687487665284876941800741376122110424479151390399847323 1;

e:=95;

RSAverify(message,N,e);
```

```
true
```

6. Building a block

Every day at 7pm, we will all collectively begin to attempt to create a block containing all the transactions from the previous 24 hours. A secret message that needs to be included in the block will be posted on the Block posting board. at 7pm in order to ensure mining does not begin until then. Whoever is the first to successfully create the block will be rewarded by being allowed to include a line that gives them an additional Choucoin.

Note that for the sake of tidiness we will use a separate board to post mined blocks. The link is here:

```
https://xoyondo.com/mb/IU4al7NzWSVW8bl
```

To build a block, the most important part is we all use the same syntax. Note that even tiny little things like a space or an extra line break will totally throw off the block building process. The syntax we will use to build a new block is:

```
BLOCKNUMBER: $12$ ⟨⟨Previous Block's Tag⟩⟩ ⟨⟨Day's Secret Message⟩⟩
⟨⟨Signed Transaction 1⟩⟩
⟨⟨Signed Transaction 2⟩⟩
⟨⟨Signed Transaction 3⟩⟩
⟨⟨Signed Transaction 4⟩⟩
```

where transactions are listed in the order they were posted (Signed Transaction 1 has an earlier time stamp than Signed Transaction 2). For example:

```
BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016
```

Indeed, the signature will usually be much longer, but regardless it should all be on a single line. Now, to build this block and get ourselves our reward of a Choucoin, we simply run the function "MineBlock()" with the inputs being your account name to receive the bounty, and the block text that we have compiled above. See Figure 5 for an example of a single attempt to mine a block. Note that the attempt fails, which is indicated by the first index entry being "false".

If instead we would like to try to mine the block repeatedly, we may use the next bit of code, an example of which is shown in Figure 6.

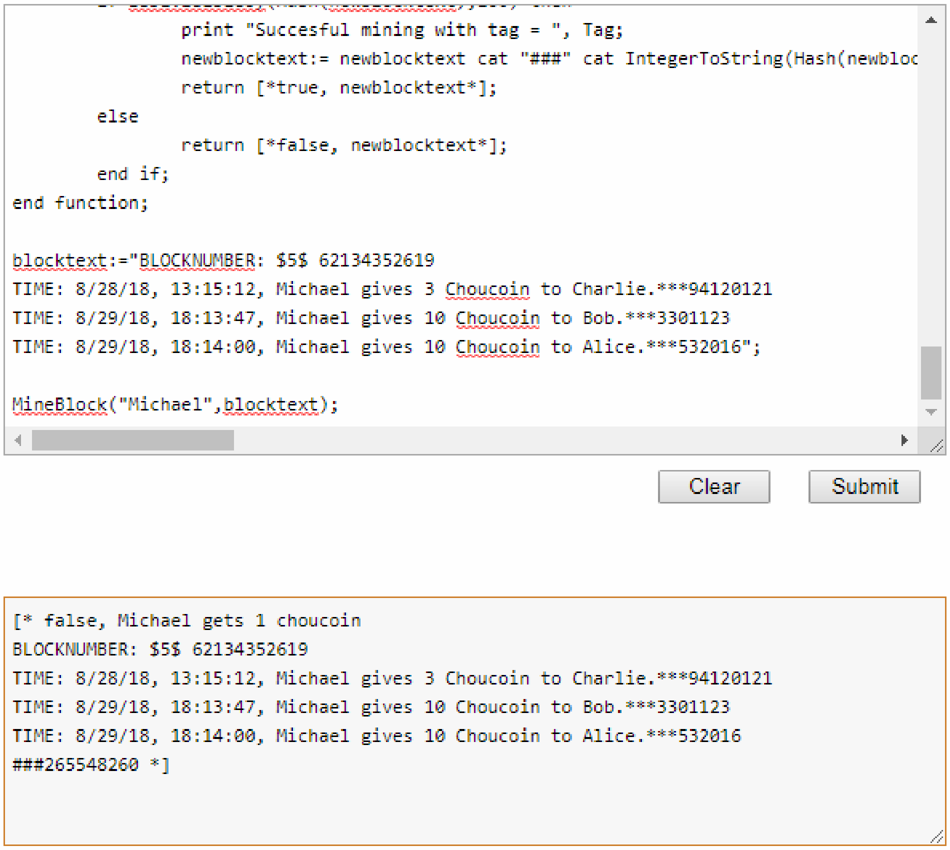
Note that when this loop succeeds, it provides the text to post to the forum to announce a new block. In this case, the text is

```
Michael gets 1 Choucoin
BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016
###6389079566###3483570000
```

This is the text that you will post to the block publishing forum:

```
https://xoyondo.com/mb/IU4al7NzWSVW8bl
```

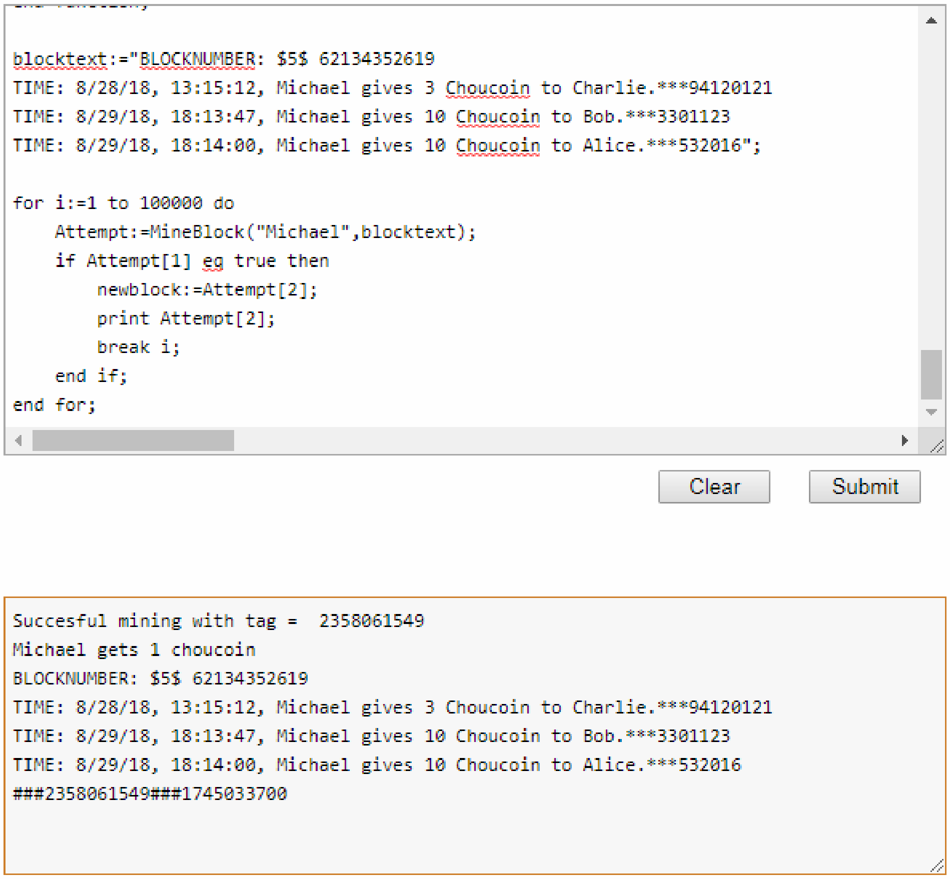
FIGURE 5. Unsuccesful mining attempt



```
                print "Succesful mining with tag = ", Tag;
                newblocktext:= newblocktext cat "###" cat IntegerToString(Hash(newbloc
                return [*true, newblocktext*];
        else
                return [*false, newblocktext*];
        end if;
end function;

blocktext:="BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016";

MineBlock("Michael",blocktext);
```

```
[* false, Michael gets 1 choucoin
BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016
###265548260 *]
```

Figure 6. Succesful mining attempt



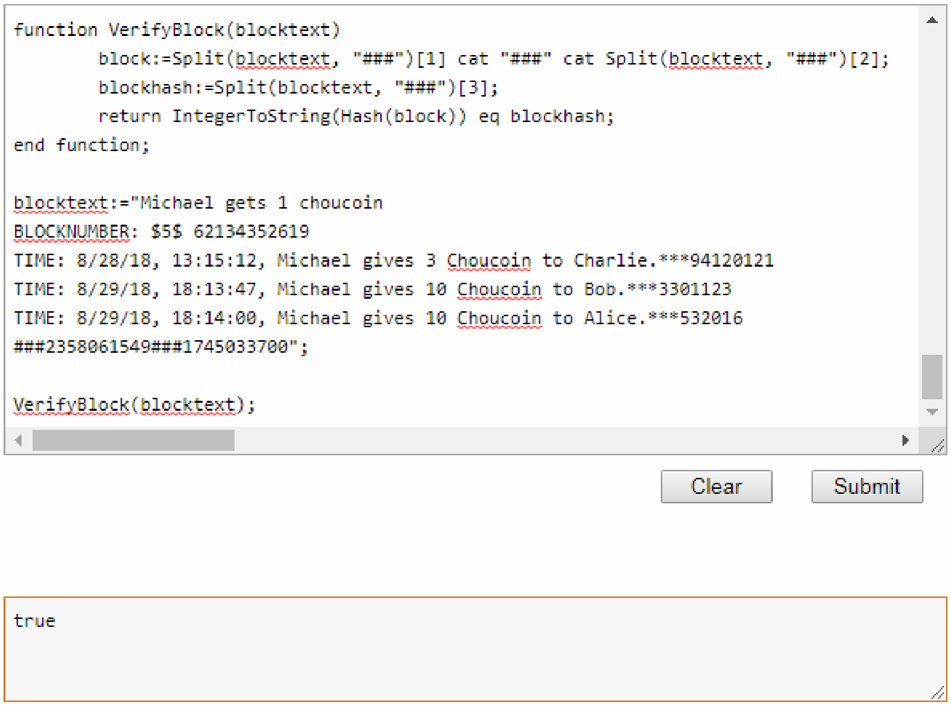
```
blocktext:="BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016";

for i:=1 to 100000 do
    Attempt:=MineBlock("Michael",blocktext);
    if Attempt[1] eq true then
        newblock:=Attempt[2];
        print Attempt[2];
        break i;
    end if;
end for;
```

```
Succesful mining with tag =  2358061549
Michael gets 1 choucoin
BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016
###2358061549###1745033700
```

7. Verifying a Block

Verifying a block is very easy, we simply copy the block's text and put it into the function "VerifyBlock()". See Figure 7 for an example. Again, just be careful that the syntax is correct; no extra spaces, enough line breaks, etc.

Figure 7. Verifying a block



```
function VerifyBlock(blocktext)
        block:=Split(blocktext, "###")[1] cat "###" cat Split(blocktext, "###")[2];
        blockhash:=Split(blocktext, "###")[3];
        return IntegerToString(Hash(block)) eq blockhash;
end function;

blocktext:="Michael gets 1 choucoin
BLOCKNUMBER: $5$ 62134352619
TIME: 8/28/18, 13:15:12, Michael gives 3 Choucoin to Charlie.***94120121
TIME: 8/29/18, 18:13:47, Michael gives 10 Choucoin to Bob.***3301123
TIME: 8/29/18, 18:14:00, Michael gives 10 Choucoin to Alice.***532016
###2358061549###1745033700";

VerifyBlock(blocktext);
```

```
true
```

References

[1] 3Blue1Brown YouTube Channel: https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw

[2] W. Bosma, J. Cannon., C. Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput., 24 (1998), 235-265.

Dept. of Mathematics, Tufts University
E-mail address: michael.chou@tufts.edu