Lulzcoin: Uma criptomoeda satírica como medidor de discussões em canais de *chat* públicos

Lucas Vieira
lucasvieira@protonmail.com
https://luksamuk.github.io

***Resumo*—Criptomoedas são uma nova fonte de investimento e de debate mundial. Propõe-se a criação da *lulzcoin*, uma criptomoeda que funciona como uma sátira à grande quantidade de novas criptomoedas que vêm surgindo atualmente, e à interação atritosa entre membros de diversas culturas em comunidades públicas online. Através das interações destes membros, calcula-se um nível de entropia dos atritos de conversas online que sirva para substituir os algoritmos de cálculo de *proof-of-work*, comumente usados em criptomoedas, gerando,**

***Index Terms*—criptomoeda, criptografia, discussões, treta, troll, bitcoin, blockchain.**

***Abstract*—Cryptocurrencies are a new worldwide source of investment and debate. We propose the creation of *lulzcoin*, a cryptocurrency which works as a satire to the great amount of new cryptocurrencies which have been appearing lately, and to the conflictuous interaction between members of several cultures in public online communities. Through the interactions of these members, we calculate an entropy level of the online chats's conflicts, which could replace the algorithms for calculation of *proof-of-work* commonly used in cryptocurrencies, therefore creating a peculiar way to mine for new blocks of a cryptocurrency.**

***Index Terms*—cryptocurrency, cryptography, discussions, flamewar, troll, bitcoin, blockchain.**

I. Introdução

Ao longo de alguns anos, criptomoedas ascenderam como uma forma de efetuar transações de forma internacional, e sem submissão a instituições financeiras centralizadas. Após a criação do *Bitcoin* [INSIRA A CITAÇÃO AQUI], diversas outras criptomoedas surgiram, cada qual com sua forma de resolver problemas identificados em sua especificação original. As criptomoedas também colocaram em evidência a estrutura de dados que existe sob as mesmas, e que dá validade às transações, de forma segura e descentralizada: a *blockchain*, uma cadeia de blocos de transações, verificadas por um sistema de pares que, ao possuir uma cópia de toda a cadeia de blocos, identificam um número que atenda a certas especificações, para que estas transações sejam adicionadas, de forma permanente, a esta cadeia, na forma de um bloco. Existem diversos tipos de especificações para a criação destes novos blocos. O mais comum é a *proof-of-work*, que sustenta a ideia da realização de um trabalho exaustivo por uma máquina, mas que seja, em contrapartida, facilmente verificável. Enquanto as implementações do *Bitcoin* utilizam o algoritmo *hashcash* [CITAÇÃO NECESSÁRIA] para implementar a *proof-of-work* de sua *blockchain*, propomos, aqui, uma nova criptomoeda, feita com o único intuito de diversão, e que tenha seu *proof-of-work* baseado na interação de usuários em aplicativos de chat ou, mais especificamente, nos atritos entre estes usuários, e na atenção em que um usuário ganha por causar este atrito. Esta característica peculiar dará forma e motor à criptomoeda aqui discutida, a *lulzcoin*. *Lulzcoin* não possui uma rigorosidade quanto ao seu formato e suas especificações como outras criptomoedas, já que sua idealização é feita, principalmente, a partir de uma brincadeira. Dado que comunidades online abertas têm a tendência a criar conflitos, devido à natureza aleatória dos indivíduos que as frequentam, a *lulzcoin* poderia beneficiar-se da interação dos membros para realizar a construção de seus blocos, tendo, portanto, pouca importância como moeda, e mais importância como um tipo de "termômetro"impreciso do calor das discussões geradas nestes ambientes.

II. Transações

III. *Proof-of-Work*

O diferencial do *lulzcoin* é a forma como funciona o seu algoritmo de *proof-of-work*. Um algoritmo desse tipo existe com o intuito de ser uma operação difícil de ser calculada em tempo hábil, mas que seja fácil verificar o resultado, quando o mesmo é obtido.

No caso do *lulzcoin*, foi idealizado um algoritmo que pudesse traduzir certos tipos de mensagens em valores de entropia e, ao fim de um ciclo, o detentor do maior valor de entropia seria o ganhador. Obviamente, este valor precisa se enquadrar à ideia do *proof-of-work* no sentido de que, mesmo sendo um tipo de generalização numérica de várias mensagens, deve ser possível identificar o recipiente que enviou a mensagem e a entropia, através de apenas um identificador.

Antes de mais nada, é preciso firmar bases para a forma como o algoritmo deverá analisar estas mensagens. A implementação primária desta tecnologia é feita com base no famoso aplicativo *Telegram* e em seu sistema de grupos. Cada mensagem de um grupo do *Telegram* pode ser uma resposta direta a outra; da mesma forma, cada mensagem pode ter também uma ou mais menções a usuários. Dados estes elementos, podemos estabelecer as seguintes regras de consideração de mensagens:

- As conversas com potencial entrópico serão chamadas *threads*. Uma *thread* é criada a partir da ligação linear entre pelo menos duas mensagens, através do sistema de resposta direta ou de menções a nomes de usuário

(`@nomedousuario`). A primeira mensagem de uma *thread* determina quem é o dono da mesma, e quem irá lucrar com a entropia gerada por esta. Mensagens subsequentes do dono da *thread* não serão consideradas para efeito de cálculos subsequentes de entropia da mesma.

- Uma *thread* pode acabar, em algum ponto, gerando "sub-*threads*", quando uma de suas mensagens recebe mais de uma resposta. Estas "sub-*threads*" são chamadas *branches*. Cada *branch* terá sua entropia calculada como uma *thread* independente e, no momento em que se tornar necessário calcular toda a entropia da *thread*, apenas a *branch* de maior altura, a partir daquele ponto, será considerada; outras serão descartadas.[1]
- Cada *thread* tem uma expectativa de vida, que aqui chamaremos de *lifetime*.[2] O *lifetime* de uma *thread* é um número que decresce com o passar do tempo; para fins de sincronização, o *lifetime* é computado antes de quaisquer operações, a partir de uma marca digital de tempo universal, que chamaremos de *timestamp*, determinando o início da thread. Dessa forma, independente das capacidades da máquina, será possível determinar o período de tempo de atividade de uma thread. O tempo de vida final, quando requerido, será a diferença de tempo entre os *timestamps* da primeira e da última mensagem da *thread*.
- Caso uma mensagem aparente ser elegível à incorporação em uma *thread* existente (por exemplo, se esta mensagem for uma resposta direta à última mensagem de uma *thread*), mas a *thread* em questão já esteja com seu *lifetime* expirado, a mensagem não será adicionada à *thread*. Esta mensagem, porém, poderá ser o início de sua própria *thread*, desde que receba uma mensagem direta.
- Estas *threads* estarão sempre agrupadas para análise, de forma que, ao fim de um certo tempo, as *threads* serão analisadas, e será agraciado o dono da *thread* de maior entropia. Cada ciclo de início de *threads*, encerrado com esta comparação de entropias, será considerado uma onda (*wave*).[3] Os tempos de vida das *waves* serão tempos fixos, reajustados através do balanceamento do nível de demanda por blocos da rede, o que será determinado utilizando pares remotos (explicado mais adiante). Cada participante só poderá ter UMA *thread* registrada por *wave*.[4]

IV. Rede

A. Consenso

V. Incentivos

VI. Constantes e Cálculos

VII. Métodos de Interação

A. Bot para Telegram

B. Carteira virtual

VIII. Conclusão

Acknowledgment

The authors would like to thank...

Referências

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[1] Talvez seja mais conveniente considerar apenas a primeira mensagem que chegar, descartando demais *branches* futuras. De qualquer forma, a própria detecção de uma *branch* precisa ser algo feito cuidadosamente, já que implica em alguma forma eficiente e não-destrutiva de análise de existência de branches. Portanto, é provável que uma decisão assim não seja necessariamente uma estratégia de simplificação de implementação.

[2] Embora não pareça tão óbvio, o *lifetime* de uma thread é um dos recursos utilizados para conferir certo alívio ao aplicativo de análise de mensagens, já que obriga *threads* a serem descartadas após certo tempo. Dessa forma, podemos abrir espaço de memória e processamento para a análise de elegibilidade de outras e de novas *threads*.

[3] Assim como o sistema de *lifetimes*, o sistema de *waves* também funciona como um dos reguladores, evitando sobrecargas no hardware minerador, e igualmente evitando monopólio de um participante sobre os outros, através do aumento de chances de lucro a partir da criação de múltiplas *threads*.

[4] Afim de minimizar inutilização de uma *thread* em andamento, recomendamos desconsiderar *threads* subsequentes ao invés de substituir velhas por novas.