Smart Contracts como uma plataforma para computação segura

Bianca Cristina da Silva, Ivan da Silva Sendin

[1]Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Uberlândia – MG – Brazil

biancacristinasilva@ufu.br, sendin@ufu.br

Abstract. *Smart contracts represent new possibilities for applications, with e-commerce and decentralized financial organizations being examples of these appplications which are able to obtain benefits from the confidence of the correct execution of programs provided by this new techonology. However, despite leading to advantagens in terms of correctness, smart contracts still produce loss of data privacy since they use a decentralized structure and this fact make the data accessible to all nodes. In this study, we present the use of multipart secure computing in the decentralized environment offered by smart contracts.*

Resumo. *Os smart contracts representam novas possibilidades de aplicações, sendo o comércio eletrônico e organizações financeiras descentralizadas exemplos dessas aplicações, as quais são capazes de obter benefícios da confiança da correta execução de programas fornecidas por essa nova tecnologia. Todavia, ainda que vantajoso no aspecto de corretude, smart contracts sofrem com a perda de privacidade dos dados, pois, uma vez que utilizam uma estrutura descentralizada, os dados são acessíveis a todos os nós da rede. Nesse estudo, apresentamos o uso da computação segura multiparte no ambiente descentralizado oferecido pelos smart contracts.*

1. Introdução

Redes descentralizadas permitem que informações sejam distribuídas e compartilhadas onde são necessárias por meio de ações dos grupos de usuários, também denominados nós, que a utilizam [Dabek et al. 2004]. Em virtude disso, infraestruturas físicas complexas tornam-se desnecessárias visto que o poder computacional da rede não concentra-se em uma única estrutura, algo que diminui a possibilidade de ocorrer falhas em um único ponto central [Petkanics 2016]. As expectativas de crescimento de redes descentralizadas são tamanhas que o diretor executivo do Twitter anunciou o projeto *Bluesky*, o qual visa desenvolver um padrão *open source* para redes sociais descentralizadas [Palmer 2019].

Atualmente, existem diversos exemplos de estruturas descentralizadas, sendo Bitcoin e Ethereum dois dos exemplos de maior sucesso nesse quesito, sobretudo, em relação a plataformas *blockchain*. Nesse estudo, a Ethereum será abordada em conjunto aos *smart contracts* como uma plataforma segura para computação multiparte. Essa escolha está relacionada ao fato de que com *smart contracts* é possível estabelecer acordos entre múltiplas entidades sem a necessidade de intermediários [Narayanan et al. 2016].

Além disso, outra vantagem associada a Ethereum é o incentivo gerado pela plataforma aos mineradores de blocos. Isso ocorre por meio do *Proof-Of-Work (PoW)*, o

algoritmo de consenso adotado pela Ethereum, no qual cada minerador recebe recompensas financeiras de acordo com o poder computacional disponibilizado para a mineração [Niu and Feng 2019]. Em virtude disso, os mineradores são incentivados a seguir as normas do protocolo, dado que isso influencia diretamente nos lucros obtidos com a Ethereum. Os valores recebidos pelos mineradores são obtidos pelas taxas pagas pelos usuários que utilizam a plataforma, sendo a unidade monetária dessas taxas denominada gás.

Todavia, para garantir o estado e corretude da estrutura descentralizada, os *smart contracts* herdam características inerentes da *blockchain*, as quais não são desejáveis em termos de computação segura. Uma dessas características é o fato dos dados armazenados na *blockchain* serem públicos, o que inviabiliza o armazenamento de informações sensíveis [Cheng et al. 2019]. Além disso, uma vez armazenados na *blockchain*, *smart contracts* são públicos e existem plataformas, tais como a *Nodesmith* e *Open Ethereum*, que permitem analisar, pesquisar e acessar *smart contracts*. Tal vulerabilidade é contornada pelo protocolo apresentado nesse estudo e será retomada em seções posteriores.

Alternativamente ao protocolo proposto por nossa pesquisa, é possível utilizar provas de conhecimento zero para resolver o problema relacionado a perda de privacidade de dados. Um desses métodos é o *Zero-knowledge Range Proofs (ZKRP)*, abordado em [Koens et al. 2017], no qual múltiplas entidades são capazes de provar que um determinado dado secreto está incluso em um intervalo. Outra forma de utilizar provas de conhecimento zero para solucionar o problema é adotar o *Zero-knowledge succinct non interactive arguments of knowledge (zk-SNARK)*, uma vez que esse método utiliza definições criptográficas e uma configuração inicial, também conhecida como *setup*, nas quais propriedades matemáticas agregagadas a uma *Common Reference String (CRS)* são usadas para garantir que as entidades envolvidas no processo não obtenham informações cruciais desnecessárias ou de forma maliciosa [Petkus 2019].

Entretanto, como apresentado em [Ben-Sasson et al. 2013], provas dessa categoria demandam esforço para gerar as chaves públicas necessárias no processo da computação e verificação das informações. De modo geral, provas de conhecimento zero representam soluções interessantes, porém com custos significativos, algo que pode ser constatado nas análises comparativas das pesquisas [Maller et al. 2019] e [Kosba et al. 2020].

2. Trabalhos Relacionados

Pesquisas semelhantes já foram desenvolvidas para solucionar problemas de segurança relacionados à computação envolvendo múltiplas entidades. O estudo apresentado em [Owoh and Mahinderjit Singh 2019], por exemplo, utiliza *smart contracts* e curvas elípticas associadas ao protocolo Diffie-Hellman para resolver o acordo de chaves entre o cliente e o nó da *blockchain*. Para atingir tal objetivo, um *framework* é proposto, no qual o acordo de chaves é obtido por meio das curvas elípticas Diffie-Hellman e os dados dos sensores são cifrados de modo que não são transmitidos em texto puro nos *smart contracts*.

Outra pesquisa relevante acerca desse tema é a desenvolvida em [Muth and Tschorsch], a qual apresenta o SmartDHX como solução para estabelecer chaves secretas a serem cifradas na sequência sem a necessidade de transmitir a chave

em si em canais não seguros como a *blockchain*. Nesse aspecto, a solução proposta beneficia-se do fato de que o problema do logaritmo discreto para a troca de chaves Diffie-Hellman, também conhecida como *Diffie-Hellman Key Exchange (DHKE)*, é considerado difícil [Paar and Pelzl 2010]. Portanto, ao utilizar *DHKE*, o problema do logaritmo discreto também é difícil para o SmartDHX.

Especificamente para computação multiparte, a pesquisa [Roman and Vu 2018] apresenta uma arquitetura que possibilita o uso de computação multiparte e *smart contracts* entre um comprador e um vendedor, por exemplo. A ideia da arquitetura proposta é combinar a característica da computação multiparte, na qual apenas o proprietário de uma informação consegue acessá-la ainda que haja interação entre múltiplas entidades, e *smart contracts*, nos quais as regras de negócio e eventos de transação estarão presentes.

3. Computação Multiparte

A computação multiparte consiste em realizar o cálculo de uma dada função utilizando poder computacional de múltiplas entidades, sendo que cada entidade possui dados privados a serem considerados como parâmetro da função [Damgård et al. 2012]. Contudo, para garantir que a função seja calculada da maneira correta e que não haja vazamento de dados privados de cada parte, é necessário desenvolver um protocolo de computação multiparte seguro.

É possível elaborar um protocolo de computação segura de diversas formas. Uma delas é apresentada em [Damgård et al. 2012], no qual utiliza-se criptografia homomórfica e provas de conhecimento zero para efetuar os cálculos de uma função entre múltiplas entidades. Outra maneira de implementar computação multiparte garantindo a segurança dos dados é utilizar algoritmos de aproximação para que uma ou mais partes consigam avaliar uma determinada função baseada nos dados de entrada privados de cada entidade envolvida, tal método foi apresentado em [Feigenbaum et al. 2006].

Nesse artigo, apresentamos uma implementação de computação multiparte baseada na pesquisa [Chaum et al. 1988], onde um protocolo é proposto a fim de permitir que múltiplas entidades sejam capazes de efetuar o cálculo final de uma dada tabela-verdade. Para atingir os resultados esperados, o protocolo garante as seguintes propriedades:

> **Dados privados não devem ser revelados.** Nesse contexto, os participantes envolvidos no processo estão seguros quando o protocolo apresentado é utilizado;
> **Custo linear em termos de número de portas e participantes.** Logo, desde que o circuito seja pequeno e a quantidade de entidades participantes restrita, o protocolo é viável em termos de custo;
> **Verificável.** Os participantes são capazes de verificar os resultados apresentados pelas outras partes envolvidas na computação;
> **Seguro contra conspiração entre participantes**. Ainda que $n - 1$ entidades conspirem entre si de forma desonesta, as informações dos participantes restantes estarão seguras.

Para garantir essas propriedades, supondo que duas entidades (A e B) desejam realizar a computação multiparte, nosso trabalho propõe a divisão do protocolo definido em [Chaum et al. 1988] em duas etapas principais: máscara e execução. A etapa de máscara refere-se aos passos que permitem que cada entidade esconda os valores originais da tabela-verdade, os quais ocorrem da seguinte maneira:

1. Uma permutação aleatória é efetuada nas linhas da tabela de entrada;
2. Mascara-se colunas pré determinadas da tabela utilizando a operação XOR e *bits* gerados aleatóriamente.

Essa etapa de máscara é efetuada localmente no computador de cada um dos participantes e gera uma tabela com a entrada ”mascarada”(figura 1).

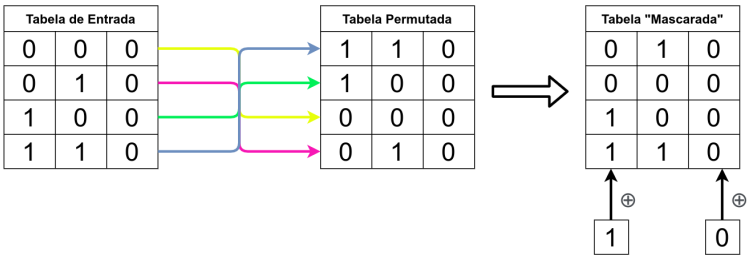

Figura 1. Etapa de máscara. A tabela-verdade sofre uma rotação nas linhas, enquanto as colunas do usuário e final são alteradas utilizando a operação XOR.

Em seguida, dado uma tabela-verdade inicial T, a computação multiparte de fato ocorre da seguinte maneira:

1. A realiza a etapa de máscara usando a tabela T como entrada e gera T' como resultado;
2. A envia T' para o *smart contract*;
3. B recebe T' por intermédio do *smart contract*;
4. B realiza a etapa de máscara usando a tabela T' como entrada e gera T'' como resultado;
5. A recebe T'' por intermédio do *smart contract*;
6. Tanto A quanto B escolhem linhas de suas respectivas tabelas bem como os bits utilizados para mascarar as tabelas e enviam essas informações para o *smart contract*;
7. O *smart contract* calcula o resultado final da computação.

Seguindo tal método a computação é efetuada, isto é, A e B obtêm o resultado da computação sem que as entradas da função computada sejam reveladas. As figuras 2 e 3 ilustram os passos descritos anteriormente.

4. SolSMC

Implementamos um *smart contract* como prova de conceito para o método de computação segura proposto em [Chaum et al. 1988]. Nesse contrato, as partes A e B podem executar uma computação sobre os seus dados sem revelá-los nem para outra parte nem para uma terceira parte.

Foi utilizada a linguagem `Solidity` para implementação do contrato e o ambiente Brownie[1]/Ganache[2] para interação do cliente com a aplicação descentralizada. O

[1] `https://github.com/eth-brownie/brownie`
[2] `https://www.trufflesuite.com/ganache`

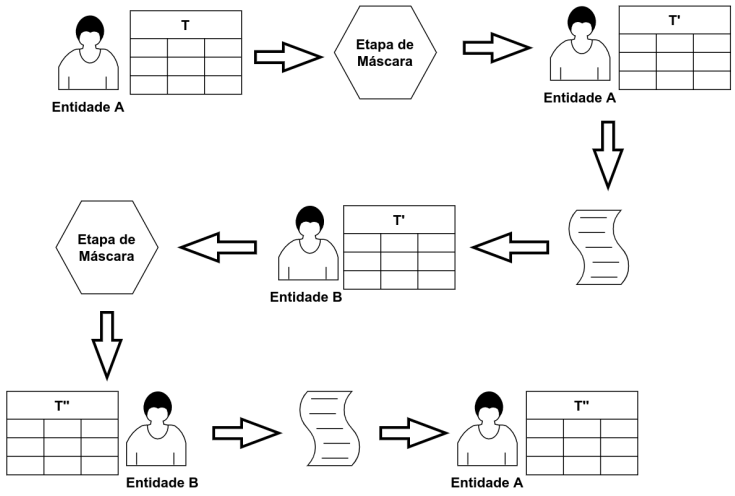

Figura 2. Etapa de execução. Os participantes executam a etapa de máscara e envio da tabela-verdade para o *smart contract* de forma sequencial.

uso do pacote Brownie/Ganache permite que o desenvolvimento e testes sejam executados localmente sem a necessidade de interação com a rede Ethereum e custos financeiros associados.

A utilização do contrato é direta e segue os passos especificados pelo método descrito anteriormente. O contrato age como uma plataforma para a troca de dados: as partes trocam as tabelas alteradas e o contrato executa a computação final revelando o resultado. Uma visão geral da execução é descrita na figura 4.

O gasto de gás da execução do contrato foi de 710812 para A e de 208592 para B. O custo total da execução do contrato é de aproximadamente ETH 0.0358568, algo que gera um valor próximo de R$51[3].

A implementação da prova de conceito está disponível em `https://github.com/BiancaCristina/Artigo-SC`.

5. Conclusões

As soluções relacionadas à *blockchain* que possibiltam execução de códigos representam novas possiblidades, visto que fornecem uma plataforma para execução de protocolos criptográficos anteriormente muito complexos para diversos cenários reais. Neste trabalho, apresentamos uma prova de conceito que explora as possibilidades proporcionadas pelos *smart contracts* implementando as funcionalidades necessárias para a computação privada envolvendo duas entidades.

[3]Valores calulados com o auxílio da ferramenta disponível em `https://ethgasstation.info`

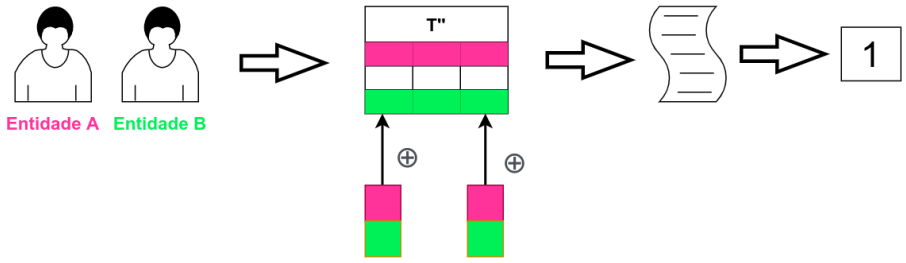

Figura 3. Etapa de execução (computação final). Ao final do processo, os participantes determinam conjuntamente a linha resultante e revelam as informações necessárias para o resultado final ser calculado.

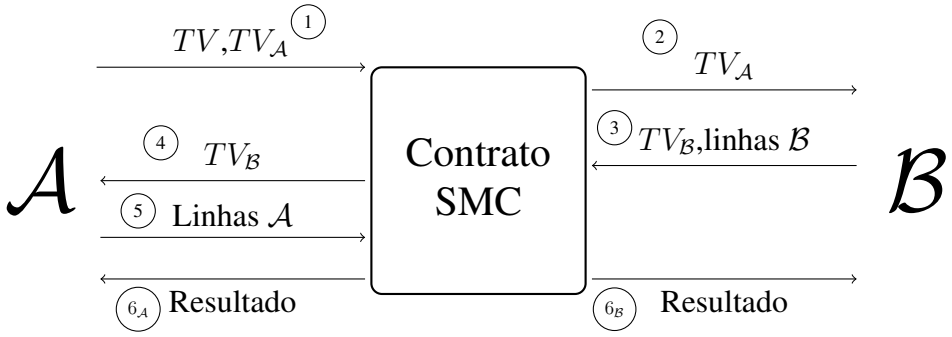

Figura 4. A iteração dos participantes com o contrato visando uma computação segura.

A solução apresentada ainda precisa aderir aos quesitos de segurança necessários, implementando os compromissos, verificações e punições para comportamento desonesto. Uma ampliação da aplicabilidade da proposta também está planejada com o aumento do tamanho da entrada da tabela verdade, permitindo a execução de computações mais complexas.

Referências

Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., and Virza, M. (2013). Snarks for c: Verifying program executions succinctly and in zero knowledge. In Canetti, R. and Garay, J. A., editors, *Advances in Cryptology – CRYPTO 2013*, pages 90–108, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chaum, D., Damgård, I. B., and van de Graaf, J. (1988). Multiparty computations ensuring privacy of each party's input and correctness of the result. In Pomerance, C., editor, *Advances in Cryptology — CRYPTO '87*, pages 87–119, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N., Juels, A., Miller, A., and Song, D. (2019). Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. *2019 IEEE European Symposium on Security and Privacy (EuroSP)*.

Dabek, F., Cox, R., Kaashoek, F., and Morris, R. (2004). Vivaldi: A decentralized network coordinate system. *ACM SIGCOMM Computer Communication Review*, 34.

Damgård, I., Pastro, V., Smart, N., and Zakarias, S. (2012). Multiparty computation from somewhat homomorphic encryption. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, page 643–662, Berlin, Heidelberg. Springer-Verlag.

Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M. J., and Wright, R. N. (2006). Secure multiparty computation of approximations. *ACM Trans. Algorithms*, 2(3):435–472.

Koens, T., Ramaekers, C., and van Wijk, C. (2017). Efficient Zero-Knowledge Range Proofs in Ethereum.

Kosba, A., Papadopoulos, D., Papamanthou, C., and Song, D. (2020). Mirage: Succinct arguments for randomized algorithms with applications to universal zk-snarks. Cryptology ePrint Archive, Report 2020/278. `https://eprint.iacr.org/2020/278`.

Maller, M., Bowe, S., Kohlweiss, M., and Meiklejohn, S. (2019). Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2111–2128, New York, NY, USA. Association for Computing Machinery.

Muth, R. and Tschorsch, F. SmartDHX : Diffie – Hellman Key Exchange with Smart Contracts.

Narayanan, A., Miller, A., Han, S., and Bailis, P. (2016). Research for practice: Cryptocurrencies, blockchains, and smart contracts; hardware for deep learning. *Queue*, 14(6):43–55.

Niu, J. and Feng, C. (2019). Selfish mining in ethereum.

Owoh, N. and Mahinderjit Singh, M. M. (2019). Applying diffie-hellman algorithm to solve the key agreement problem in mobile blockchain-based sensing applications. *International Journal of Advanced Computer Science and Applications*, 10.

Paar, C. and Pelzl, J. (2010). *Public-Key Cryptosystems Based on the Discrete Logarithm Problem*, pages 205–238. Springer Berlin Heidelberg, Berlin, Heidelberg.

Palmer, A. (2019). *Twitter CEO Jack Dorsey has an idealistic vision for the future of social media and is funding a small team to chase it*.

Petkanics, D. (2016). *The Benefits of Decentralization*.

Petkus, M. (2019). Why and how zk-snark works. *CoRR*, abs/1906.07221.

Roman, D. and Vu, K. (2018). Enabling data markets using smart contracts and multiparty computation. pages 258–263.