Towards to fair trade with item validation using Bloom Filters and Smart Contracts

Matheus Cunha Reis, Ivan da Silva Sendin

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

matheuscunhareis30@gmail.com, sendin@ufu.br

Abstract. *Smart Contracts bring a new set of possibilities in the secure protocol development: participants gain guarantees of the correct execution of the protocol steps. In contrast, current Smart Contract solutions do not provide data privacy, this scenario is particularly bad in the commerce of* e-goods*. In this work, we present a fair trade protocol based Smart Contract and Bloom Filter for the problem of e-goods trading where the involved parts do not trust each other.*

1. Introduction

A protocol is called fair if at the end of its execution, all participants receives the previously agreed or none of them receives anything. Fair protocols gained importance due to the need to establish security in the online sales of *e-goods*, in this scenario it is easy to see that the part who takes the first action - delivers the product or makes the payment - is at a disadvantage compared to the other [4, 20]. The fairness can be achived using Trusted Thirty Party as a intermediary, gradual release or optimistic protocols [3, 18]. Another recurring problem in the *e-goods* commerce is item validation problem: the merchant can deliver the product - a bit string - but it is difficult to establish whether the delivered product is really what the buyer expected [21].

This work addresses the problem where two parts not mutually trustworthy want to negotiate the sale of information - represented by a set of strings - in a secure manner, that is, the buyer will receive the information for which he actually paid for and the seller will receive the money for the information. This scenario arises, for example, when $\mathcal{A}$ and $\mathcal{B}$ are working independently in a large database, selecting items using some pre-established criteria, for example selecting symptoms from a set of medical records or Blockchain addresses with specified properties. After a while, each one is expected to produce their own list with a non-empty intersection and someone want to buy the items and pay only for the new items.

1.1. Smart Contracts

Introduced by Szabo in [22], **Smart Contracts** (SC) are programs that we trust in their correct execution. Currently, the Ethereum ecosystem is the main platform for the development and execution of SC[23]. Shortly, the desired security properties of smart contracts are obtained with the redundancy of the execution of the SC on nodes of a P2P network. Each participating node receive an economic incentive for the correct execution of the contracts. The SC owner pays a fee - called *gas* on Ethereum - for the execution of the

contract. This fee depends on the computational power - both processing and memory - necessary to execute the contract.

This approach causes a loss of confidentiality, both in the SC itself and in the data: all nodes in the P2P network have access to data and code [1].

1.2. Bloom Filters

Bloom Filter[8] is a data structure that implements the *add* and *contains* methods; the Bloom Filter is probabilistic: sometimes the *contains* method produces false positive.

The backend of a Bloom Filter is a bit vector with size n; to add some element e to a filter, a family of k cryptographic hash functions[2] is used set on k bits of the backend vector. In Figure 1 is presented a Bloom Filter containing two elements - **A** and **B**. When someone query this BloomFilter for **C**, a false positive occurs. Given expected number of elements and a tolerable probability of false positives, the parameters k and n can be determined.

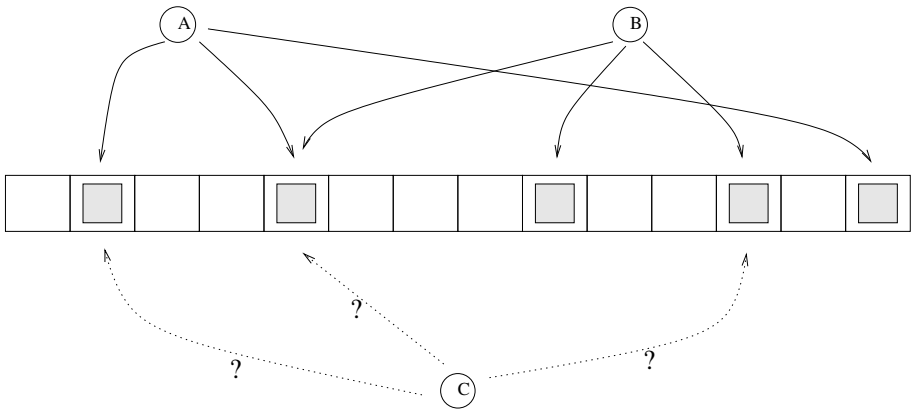

Figure 1. A Bloom Filter with n=14 and k=3.The elements A and B were included in this Bloom Filter. Consulting if C is in this Bloom Filter results a false positive.

Bloom filters are space-efficient and have some powerful properties like:

- **Set Union:** Given two filter with the same size and using the same family of hash functions, the set union operation is given by the bitwise OR operation;
- **Set Intersection:** Similarly, the set intersection is obtained by bitwise AND;
- **Set Size:** The number of elements added to filter can be calculated approximately by this formula given X, the number of bits '1' in the filter:

$$S = -\frac{n}{k} - \ln(1 - \frac{X}{n}).$$

Although bloom filters are built using cryptographic hash functions with properties of unidirectionality and second pre-image resistance, bloom filters offer very limited privacy in some scenarios: given a Bloom Filter determining a x that is in the filter has

[1] The data from SC can also be easily obtained using services offered by services such as `etherscan.io`.

[2] Cryptographic hash functions produces hash codes not suitable to be used directly as index to an array, usually just some bits of hash code is used.

difficulty limited by the probability of false positives of the filter. In the context of a "dictionary attack", an opponent can easily brute force the dictionary and test each element. In some scenarios, the privacy properties are enough and Bloom Filter are used in some cryptocurrency protocols (see [10] for example).

Some authors propose advances in Bloom Filters to provide better privacy. In [17] the authors propose the partitioning of the filter in order to obtain privacy. The use of cryptographic schemes based on Pohlig-Hellman Encryption [6], and the use of Blind Signatures and Oblivious Pseudorandom Function [19] are also proposed.

2. Related Work

Recently, the cryptocurrency ecosystems has attracted attention and several fair protocols have been developed using the capabilities of Blockchain.

Several protocols use advanced features of Bitcoin transactions to develop Smart Contract that implement fair protocols.

In [12, 15, 7, 16, 5] some protocols are proposed to achieve secure multi-party computation. In [9] the authors propose a protocol for signing contracts: after the participants agree on the contract, if someone give up from signing, he suffer a financial penalty. Also, in [13] a fair protocol which anonymity for both buyer and merchant is presented.

3. Fair Trade Protocol

In this section we present the `FairContract` protocol that uses Bloom Filters and Smart Contracts to provide a fair trade with item validation. The protocol is optimistic: the role of the contract is minimal when the involved parties behave honestly.

3.1. Preparation

As stated before, an entity $\mathcal{A}$ owning a list of words $L_{\mathcal{A}}$ wants to buy from $\mathcal{B}$ only "new words" of $L_{\mathcal{B}}$. $\mathcal{A}$ and $\mathcal{B}$ agree with a Bloom Filter specification and a initial collateral to be paid in case of misbehavior. After that, each one produces in private their respectives filters: $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$.

$\mathcal{A}$ creates and deploy the `FairContract` and send to it a commitment $Commit_{\mathcal{A}}$ for $BF_{\mathcal{A}}$ and a previously accorded collateral $\$_{\mathcal{A}}^{Col}$. After $\mathcal{B}$ send his commitment $Commit_{\mathcal{B}}$ and the collateral $\$_{\mathcal{B}}^{Col}$(step 2 in Figure 2) $\mathcal{A}$ sends $BF_{\mathcal{A}}$ and get the collateral back. This step of the protocol is symmetrical, so $\mathcal{B}$ acts in the same way. Both participants are compelled to act honestly with the penalty of losing the collateral or the privacy of $L_{\mathcal{B}}$.

Figure 2. `FairContract` **preparation.**

At this point, both participants known $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ and approximately calculate the size of the set of new words

$$|L_{\mathcal{B}}| - |L_{\mathcal{A}} \cap L_{\mathcal{B}}|.$$

The bulk computations performed in this phase are private, each entity performs the computation on your own equipment. The off-chain comunication can be executed using traditional secure means[3].

An important feature about using Bloom Filters in this protocol is that it allows one party to verify the honesty of the other party even before the data or values change take place. Using $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$, $\mathcal{A}$ can estimate the size of $|L_{\mathcal{A}} \cap L_{\mathcal{B}}|$, and in this way have some thoughts about the "quality" of $L_{\mathcal{B}}$: if the size of estimated intersection is too small, $\mathcal{A}$ concludes that $\mathcal{B}$ created his filter with random words and gives up on buying $L_{\mathcal{B}}$.

At this point, $\mathcal{B}$ creates a subset $L^r_{\mathcal{B}}$ from $L_{\mathcal{B}}$ such that the BF_r filter created from $L^r_{\mathcal{B}}$ have the following property:

$$BF_{\mathcal{A}} \vee BF_{\mathcal{B}} = BF_{\mathcal{A}} \vee BF_r,$$

where $\vee$ stands for bitwise `or` operator.

$L^r_{\mathcal{B}}$ is the list with the items that $\mathcal{A}$ is buying, $\mathcal{B}$ creates this list for two main reasons:

- In case $\mathcal{A}$ just switch to "1" some bits from $BF_{\mathcal{A}}$- just to pay less to $\mathcal{B}$- $\mathcal{A}$ would get no bennefits from this once it gets proportionally fewer items;
- In the case of litigation, less gas will be spent (see Table 1).

Ideally, $L^r_{\mathcal{B}}$ should be the minimal set that meets the restriction of covering. Finding such minimum subset of coverage is NP-Hard[14]. As $L^r_{\mathcal{B}}$ does not impact the correct execution of the protocol, only at a possible cost, a greedy algorithm is sufficient.

3.2. Protocol Execution

Using off-chain communication, $\mathcal{A}$ and $\mathcal{B}$ agree on the following values:

- $\$_{\mathcal{A}}$: to payment for $L_{\mathcal{B}}$;
- $v_{\mathcal{B}}$**:** a collateral to be deposited by $\mathcal{B}$;
- α**:** a value to define a penalty factor to be paid according with the situation (see Table 1).

The steps to implement the protocol are presented next, first of the honest behavior of the participants and then of the scenarios of possible misbehavior.

3.2.1. Honest Execution

Whenever $\mathcal{A}$ and $\mathcal{B}$ behave honestly, the protocol is straightforward:

[3]Secure e-mail or instant messaging.

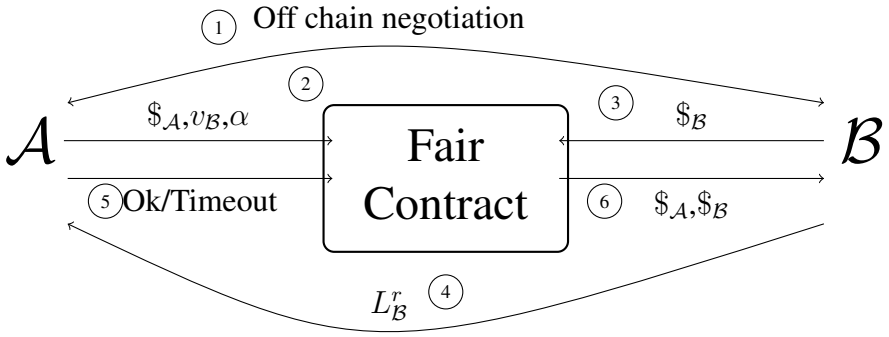

Figure 3. `FairContract` **execution for honest** $\mathcal{A}$ **and** $\mathcal{B}$**.**

- $\mathcal{A}$ sends $\$_{\mathcal{A}}$, the agreed collateral value $v_{\mathcal{B}}$ and α factor;
- $\mathcal{B}$ binds to `FairContract` sending $\$_{\mathcal{B}}$, corresponding to $v_{\mathcal{B}}$;
- $\mathcal{B}$ sends $L^r_{\mathcal{B}}$ to $\mathcal{A}$, this step is off chain;
- $\mathcal{A}$ verifies $L^r_{\mathcal{B}}$, if $L^r_{\mathcal{B}}$ corresponds to previously agreed $BF_{\mathcal{B}}$, $\mathcal{A}$ sends a "sale accepted" message to contract, that allows $\mathcal{B}$ withdraw the payment($\$_{\mathcal{A}}$) and the collateral($\$_{\mathcal{B}}$), finishing the `FairContract`.

The steps for honest behavior are described in Figure 3.

3.2.2. $\mathcal{B}$ Acting Dishonestly

If $\mathcal{B}$ does not send the correct $L^r_{\mathcal{B}}$, $\mathcal{A}$ activates the contract's *litigious* mode (Figure 4). This action involves two alternatives scenarios:

1. $\mathcal{B}$ sends $L^r_{\mathcal{B}}$ to `FairContract`, once $L^r_{\mathcal{B}}$ is validated using $BF_{\mathcal{B}}$, $\mathcal{B}$ receives $\$_{\mathcal{A}}$ and $\alpha\$_{\mathcal{B}}$ (Steps 1,2 and 3 in Figure 4);
2. If after a certain amount of time that the litigious mode is activated $\mathcal{B}$ does not send $L^r_{\mathcal{B}}$ to the `FairContract`, $\mathcal{A}$ invokes `FairContract` and receives $\$_{\mathcal{A}}$ and $\$_{\mathcal{B}}$, penalizing $\mathcal{B}$(Steps 1 and $2'$ in Figure 4).

3.2.3. $\mathcal{A}$ Acting Dishonestly

Even receiving $L^r_{\mathcal{B}}$, $\mathcal{A}$ can trigger the litigious mode of the `FairContract`.

As $\mathcal{B}$ has $L^r_{\mathcal{B}}$, the list will be sent to the `FairContract`, $\mathcal{A}$ will have no financial benefit from this action and lose the confidentiality of the list just purchased.

Figure 4. `FairContract` **in litigious mode.** $\mathcal{B}$ **sends** $L^r_{\mathcal{B}}$ **to** `FairContract` **and receives the payment. If not, after a timeout,** $\mathcal{A}$ **receives the contract values.**

Dishonest Behavior	Reaction	Penalty
$\mathcal{A}$(or $\mathcal{B}$) don't open the commit for filter	`FairContract` blocks the collateral	The collateral is lost.
$\mathcal{A}$ inflates $BF_{\mathcal{A}}$ with random bits	$\mathcal{B}$ creates $L^r_{\mathcal{B}}$ based on the difference of $BF_{\mathcal{A}}$ from $BF_{\mathcal{B}}$	$\mathcal{A}$ receives less information from $\mathcal{B}$
$\mathcal{B}$ don't send $L^r_{\mathcal{B}}$	$\mathcal{A}$ put `FairContract` in litigious mode	$\mathcal{B}$ loses $\$_{\mathcal{B}}$
$\mathcal{A}$ calls litigious improperly	$\mathcal{B}$ sends $L^r_{\mathcal{B}}$ through `FairContract`	$\mathcal{A}$ loses $L^r_{\mathcal{B}}$ confidentiality $\mathcal{B}$ loses $(1-\alpha)\$_{\mathcal{B}}$
$\mathcal{B}$ "holds" $L^r_{\mathcal{B}}$	$\mathcal{A}$ calls litigious mode, $\mathcal{B}$ sends $L^r_{\mathcal{B}}$ through `FairContract`	$\mathcal{A}$ loses $L^r_{\mathcal{B}}$ confidentiality $\mathcal{B}$ loses $(1-\alpha)\$_{\mathcal{B}}$

Table 1. Summarization of dishonest behavior and penalties in `FairContract`.

4. Protocol Analysis

Below we present an empirical analysis of the contract, for the sake of simplicity we will not consider the cost of execution, the gas charged by the Ethereum platform, for reasonable prices negotiated in the contract it is expected that this cost will be negligible.

- The expensive task of creating $BF_{\mathcal{A}}$, $BF_{\mathcal{B}}$ and $L^r_{\mathcal{B}}$ is executed off chain;
- $\mathcal{A}$ can verify p7lausibility of $BF_{\mathcal{B}}$ testing if some of items from $L_{\mathcal{A}}$ are in $BF_{\mathcal{B}}$, this procedure minimizes the likelihood of $\mathcal{B}$ creating a filter with random bits;
- $\mathcal{A}$ has no gains in increasing or decreasing the $BF_{\mathcal{A}}$ using random bits;
- When the parties involved act honestly the cost of the contract is small, the bulk of data transmission and computing is executed off chain. As $L^r_{\mathcal{B}}$ does not go through the contract, its privacy is preserved;
- If $\mathcal{B}$ is unable to present the items to $\mathcal{A}$ or to `FairContract`, it loses the collateral;
- If $\mathcal{A}$ acts dishonestly, calling litigious mode even after receiving $L^r_{\mathcal{B}}$, $\mathcal{B}$ is forced to send $L^r_{\mathcal{B}}$ through the contract, causing loss of confidentiality;
- If $\mathcal{B}$ delays the release of $L^r_{\mathcal{B}}$, forcing the litigious mode, causing loss of confidentiality, $\mathcal{B}$ loses $(1-\alpha)\$_{\mathcal{B}}$.

It is important to note that the contract fails to distinguish between the scenario in which $\mathcal{A}$ acts dishonestly by calling the litigious mode improperly and the scenario in which $\mathcal{B}$ "holds" $L^r_{\mathcal{B}}$ waiting for $\mathcal{A}$ to call the litigious mode. In these two cases both participants are penalized: $\mathcal{A}$ loses $L^r_{\mathcal{B}}$ confidentiality and $\mathcal{B}$ loses (1-α)$\$_{\mathcal{B}}$. Thus, to avoid these scenarios, the values of $\$_{\mathcal{B}}$ and α need to be chosen according to an estimated value of $L^r_{\mathcal{B}}$ confidentiality. A summary of dishonest behavior is presented in Table 1.

5. Implementation

The fair protocol was implemented using the Truffle Suite (Truffle & Ganache) [11]. Truffle is the most popular Ethereum framework to create, execute and test smart contracts. It allows developers to deploy, link libraries, write automated tests and manage network artifacts with an easy and fast API. Among their usabilities, Truffle provides the communication between the tests and the deployed contracts on Ethereum networks. Ganache is a tool that creates a personal Ethereum blockchain where you can execute commands, run

tests, inspect and debug the state of contracts easier and faster than using online Ethereum networks.

In this implementation, along with Truffle Suite, Web3.js [2], a library that allows you to interact with a local or remote Ethereum node using HTTP, IPC or WebSocket, and Mocha [1], a framework for build test scenarios, helped in the creation of tests for the possible situations of contract operation. The contract and his tests are available at `https://github.com/matheuscr30/Word-Sale-Fair-Contract`.

6. Conclusions and Future Work

In this paper, we introduced a fair protocol based on Smart Contract and Bloom Filter, where the parties are encouraged to behave honestly, otherwise they suffer financial and confidentiality losses. The use of the Bloom Filter allows items to be validated at low cost before financial transactions take place.

The proposed protocol is optimistic in the sense that if the parties behave honestly, the participation of the contract is minimal. The part of the protocol that requires significant CPU costs is executed in privately by the parties involved, saving gas charged by Ethereum.

The use of regular Bloom Filter provides the necessary privacy in some scenarios. But in others - for example when an adversary can test some item in the filter instead of producing them - this approach is not enough. In future, we plan to study the viability of extending this work by using more secure alternatives to Bloom Filters.

References

[1] Mocha project. `https://mochajs.org/`, 2020. Accessed: 2020-05-22.

[2] Web3 project. `https://web3js.readthedocs.io/en/v1.2.11/`, 2020. Accessed: 2020-05-22.

[3] Abdullah Alotaibi and Hamza Aldabbas. A review of fair exchange protocols. *International Journal of Computer Networks & Communications (IJCNC)*, 4(4):307–319, July 2012.

[4] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 6–17, 1997.

[5] Adam Back and Iddo Bentov. Note on fair coin toss via bitcoin. *CoRR*, abs/1402.3698, 2014.

[6] Steven Michael Bellovin and William R. Cheswick. Privacy-Enhanced Searches Using Encrypted Bloom Filters. *Columbia University Academic Commons*, pages CUCS–034–07, 2007.

[7] Iddo Bentov and Ranjit Kumaresan. How to use Bitcoin to design fair protocols. *Lecture Notes in Computer Science (LNCS)*, 8617(PART 2):421–439, 2014.

[8] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[9] Josep Lluis Ferrer-Gomila, M. Francisca Hinarejos, and Andreu Pere Isern-Deyà. A fair contract signing protocol with blockchain support. *Electronic Commerce Research and Applications*, 36(June):100869, 2019.

[10] Arthur Gervais, Srdjan Capkun, Ghassan O. Karame, and Damian Gruber. On the privacy provisions of bloom filters in lightweight bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, page 326–335, New York, NY, USA, 2014. Association for Computing Machinery.

[11] Truffle Blockchain Group. Truffle suite. `https://www.trufflesuite.com`, 2020. Accessed: 2020-05-22.

[12] Lijuan Guo, Xuelian Li, and Juntao Gai. Multi-party Fair Exchange Protocol with Smart Contract on Bitcoin. *International Journal of Network Security*, 21(1):71–82, 2019.

[13] Danushka Jayasinghe, Konstantinos Markantonakis, and Keith Mayes. Optimistic fair-exchange with anonymity for bitcoin users. *Proceedings - 11th IEEE International Conference on E-Business Engineering, ICEBE 2014 - Including 10th Workshop on Service-Oriented Applications, Integration and Collaboration, SOAIC 2014 and 1st Workshop on E-Commerce Engineering, ECE 2014*, pages 44–51, 2014.

[14] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[15] Ranjit Kumaresan and Iddo Bentov. How to Use Bitcoin to Incentivize Correct Computations. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, pages 30–41, 2014.

[16] Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to Use Bitcoin to Play Decentralized Poker. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 195–206, 2015.

[17] Pierre K Y Lai, S M Yiu, K P Chow, C F Chong, and Lucas C K Hui. An efficient bloom filter based solution for multiparty private matching. *Proceedings of the 2006 International Conference on Security & Management, SAM*, pages 286–292, 2006.

[18] Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, PODC '03, page 12–19, New York, NY, USA, 2003. Association for Computing Machinery.

[19] Ryo Nojima and Youki Kadobayashi. Cryptographically secure bloom-filters. *Transactions on Data Privacy*, 2(2):131–139, 2009.

[20] Henning Pagnia, Holger Vogt, and Felix C. Gärtner. Fair Exchange. *The Computer Journal*, 46(1):55–75, 01 2003.

[21] F. Piva and R. Dahab. E-commerce and fair exchange - the problem of item validation. *Proceedings of the International Conference on Security and Cryptography*, pages 317–324, 2011.

[22] Nick Szabo. Formalizing and Securing Relationships on Public Networks. *First Monday*, 2(9), 1997.

[23] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.