Tweetchain: An Alternative to Blockchain for Crowd-Based Applications

Francesco Buccafurri(✉), Gianluca Lax, Serena Nicolazzo, and Antonino Nocera

DIIES, University Mediterranea of Reggio Calabria,
Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy
{bucca,lax,s.nicolazzo,a.nocera}@unirc.it

Abstract. The assurance of information in the crowdsourcing domain cannot be committed to a single party, but should be distributed over the crowd. Blockchain is an infrastructure allowing this, because transactions are broadcast to the entire community and verified by *miners*. A node (or a coalition of nodes) with high computational power can play the role of miner to verify and approve transactions by computing the *proof of work*. Miners follows a highest-fee-first-served policy, so that a provider of a Blockchain-based application has to pay a non-negligible fee per transaction, to increase the likelihood that the application proceeds. This makes Blockchain not suitable for small-value transactions often occurring in the crowdsourcing paradigm. To overcome this drawback, in this paper we propose an alternative to Blockchain, leveraging an online social network (we choose Twitter to provide a proof of concept). Our protocol works by building a meshed chain of public posts to ensure transaction security instead of proof of work, and no trustworthiness assumption is required for the social network provider.

Keywords: Crowdsourcing · Blockchain · Twitter · Public ledger

1 Introduction

Coordination, record keeping, and irrevocability of transactions are features that make the Blockchain technology exploitable not just for cryptocurrencies. Indeed, a variety of applications can be built on top of this technology, making Blockchain a registry and inventory system for the recording, tracking, monitoring, and transacting of all assets [8]. Blockchain is considered a robust platform: even though a number of possible attacks are known [1,6,7,9,10], they are mostly not relevant in practice.

Blockchain is recently attracting the interest of both companies and researchers due to its power and the possibility to implement innovative solutions in those cases in which many users are spread over large spaces and may belong to different organizations. Despite this, there are a lot of situations in which the use of Blockchain appears not suitable. Indeed, in Blockchain, the consensus on

J. Cabot et al. (Eds.): ICWE 2017, LNCS 10360, pp. 386–393, 2017.
DOI: 10.1007/978-3-319-60131-1_24

the ledger is reached thanks to the computation of the *proof of work* done by *miners*. A node (or a coalition of nodes) with high computational power can play the role of miner and follows (in general) a highest-fee-first-served policy to select transactions being verified. This implies that a provider of a Blockchain-based application has to pay a non-negligible fee per transaction, to increase the likelihood that its application proceeds. Obviously, in the case of small-value transactions often occurring in the crowdsourcing paradigm, this model appears little plausible.

There is another reason for which Blockchain seems not suitable for our setting. We expect that crowd-based applications are mostly relying on the participation of a large number of people through their mobile devices. But, any entity participating in the Blockchain system, even a simple user, must join a P2P network, and must exchange data in upload and download continuously. Moreover, a significant storage space is required, to store enough amount of past transactions to limit the need of contacting other nodes to retrieve them (how to do this in a secure way is not a trivial task). Both the above aspects are little compliant with the mobile and ubiquitous setting we refer to. Moreover, joining a P2P network, could be not well-accepted from many people, even for the perception of less security.

In this paper, we propose an alternative to Blockchain aimed at overcoming the above drawbacks and thus designed to make public-ledger-based solutions suitable to the crowdsourcing domain. Besides this domain, also Web-of-Things (WoT) applications could have benefits from our solution, because of similar motivations (battery consumption, computational and storage requirement, need of robust updates, need of mining on IoT devices, and so on).

This paper is organized as follows. In the next section, we provide a quick overview of the proposal. Then, in Sect. 3, we present our approach. We provide a preliminary security analysis in Sect. 4, and finally, we draw our conclusions in Sect. 5.

2 Overview of the Proposal

Our solution exploits the high capability of online social networks to share information among people. Besides communication media, online social networks can become part of people workflows. The advantages are reciprocal. People are used to operate on social networks profiles, there is no extra cost, availability of the social network services is very high. For social network providers, the role in the society would become more central, and this would increase business. Social networks may thus help us to overcome the limits of Blockchain in the crowdsourcing domain.

To be effective, we focus our attention on a real-life social network, namely Twitter, but in principle, we could use any social network implementing a posting mechanism and a key-based search for posts. The idea is not to implement Blockchain over Twitter, but to reinvent a consensus protocol using tweets to encode transactions and meshed replications to substitute the proof of work and thus the need of fees for miners.

We call this protocol `Tweetchain`. Transaction tweets, together with a sufficient number of confirmation tweets posted by randomly selected members of the community, result in a meshed chain of replicated transactions. Under a proper non-collusion assumption, it is impossible for any party, including Twitter itself, to delete, alter, or forge valid transactions. From a conceptual point of view the protocol is truly decentralized. Indeed, there is no a central role of some node. However, no P2P tool must be enabled on the client machine. A user of the system only must run an application working on his Twitter profile. Importantly, Twitter does not play the role of trusted third party. Moreover, it is not the provider of some part of the protocol. The role of Twitter respect the protocol is transparent. As a consequence, Twitter cannot succeed as an adversary. This conclusion is based on the presence of the meshed network of transactions, for which no selective omission in publishing transaction tweets can be done. Twitter could, in principle, only stop the entire service, that is the entire Tweetchain community. Even though we can argue that this situation is very little plausible (like the fact that Blockchain is stopped by bodies providing the Internet), our protocol does not require a single social network. It could operate over multiple social networks having the basic required features. There are also models and technologies supporting multiple-social-network approaches [2,3]. This would further reduce the chance of success of a denial of service attack.

We stress that our goal is not that of replacing Blockchain, which obviously remains preferable in all the cases, as for cryptocurrencies, in which nobody should not have the possibility to stop the service. Tweetchain can be viewed as a lightweight public ledger for non-critical services, oriented at least to the domains of crowdsourcing and WoT.

This paper is at a preliminary stage. Therefore, the security of the protocol is only argued and intuitively understood. Moreover, a more abstract model can be designed to better present the idea and support its theoretical analysis. Anyway, we think that the direction of our research is promising, for the possible application impact.

3 The Tweetchain Model

In this section, we describe our approach and the structure of the messages exchanged by the involved entities.

The main actors of our model are:

- The Twitter social network, and, in particular, the followings features:
  1. the posting of *tweets* for registered users;
  2. the notification on the *follows* activity;
  3. the searching for information by *hashtags*.
- a welcome profile W used to implement a sort of yellow page support.
- the `Tweetchain` community, namely C, of users who join the `Tweetchain` protocol.

As a prerequisite, to participate in the `Tweetchain` community, a user must be able to build, by starting from a secret, a (SHA-256 based) hash chain [5] of a given size, say k, which will be used to maintain all his timeline activities linked together. This way, as will be clearer in the following, no modification can be done on older messages without compromising the remaining part of the user timeline. The value k, representing the length of the hash chain, is a system parameter which also limits the maximum size of the chunk of the user timeline (intended as number of tweets) that must be considered to verify the validity of a given message.

All the detail on the usage of this hash chain will be clarified in the following. As a further observation, we will consider our system in a steady-state, meaning that there are always at least $s = \frac{2t}{1-m}$ members in the `Tweetchain` community, where t and m are system parameters discussed in Sect. 4.

Now, we are ready to describe our proposal. The `Tweetchain` paradigm is composed of the following protocols.

Registration. It is executed by each user, say x, who wants to become member of the `Tweetchain` community C. Clearly, a prerequisite of this protocol is the sign up to Twitter in order to create a profile on it.

The first step he performs is to follow the welcome profile W and to publish a `hello` tweet with the following structure: $\langle \#HC_x^1 \#HC_W^1$ Hello $@W \rangle$, where $\#HC_x^1$ and $\#HC_W^1$ are Twitter hashtags with the base64 encoding of the first element of the hash chain of x and W as text, respectively, and $@W$ is a Twitter reference to the welcome page W. After that, W verifies the tweet of x and sends a confirmation tweet as a `welcome` message with the reference to this user and a link to his `hello` tweet. Suppose that W has already posted $i-1$ tweets, then the `welcome` message for x will have the following structure: $\langle \#HC_W^i$ Welcome $@x \#HC_x^1 \#TID_x^1 \rangle$ where $\#HC_W^i$ and $\#HC_x^1$ are Twitter hashtags of the base64 encoding of the ith element of the hash chain of W and the first element of the hash chain of x, respectively, $@x$ is a Twitter reference to the user x, and $\#TID_x^1$ is a Twitter hashtag with the ID of the first tweet (`hello`) of x as text. Observe that, the ID of a tweet (or status ID) is always unique inside Twitter.

As a consequence, W contains at least one tweet for each member of the community in join-chronological order. After this, x generates at random the set F_x of followings who will validate his transactions in the future. This set is built as follow:

- x retrieves his Twitter identifier. (Recall that each Twitter user has a unique 64-bit numeric identifier.)
- Then, this identifier is used as seed of a community-known PRNG to extract s random numbers and for each number, say n, computes $n \ mod \ w$, where w is the total number of tweets posted by W (i.e., the size of the `Tweetchain` community).
- At this point, the numbers computed above are used as indexes to select s distinct screen names from the welcome profile W.
- x sends a *private* message to each of the s profiles, whose screen names have been derived in the previous step, to ask them to follow him.

– After verifying the legitimacy of the request of x by using the community PRNG, each of the profiles contacted by x adds a follow link towards x and duplicates the `welcome` tweet of W by replacing $\#HC_W^i$ with their current hash chain element.

Transaction generation. This protocol allows the generation of a new transaction. Similarly to what happens in Blockchain, each transaction carries different information, such as: *(i) the timestamp of the generation,* (ii) a content, i.e., the transaction payload, *(iii) an input transaction, and* (iv) a target profile acting as transaction recipient.

In our protocol, the generation of a new transaction is associated with that of a new *tweet* by the user, in the following referred as *t-tweet*. According to the requirements described above, the i-th *t-tweet* of the user x, will have the following structure: $\langle \#HC_x^i \#TID_y^p$ content $@r\rangle$, where $\#HC_x^i$ is a hashtag with the base64 encoding of the ith element of the hash chain of x, $\#TID_y^p$ is a hashtag of the ID of the p-th tweet posted by the user y and used as input for this transaction, and $@r$ is a Twitter reference to the recipient r of this transaction.

As soon as x posts a new *t-tweet*, the s users of F_x following him will be notified by the Twitter platform automatically. They will proceed by verifying the legitimacy of this new transaction by using the verification protocol described below. After running the verification procedure, they will publish a confirmation tweet on their timeline. Now, let v be one of the users of F_x and let $j-1$ be the number of tweets generated by v till now, his confirmation tweet for the transaction of x will be: $\langle \#HC_v^j @x \#TD_x^i$ status $\#TID_y^p$ content $@r\langle$, where $\#HC_v^j$ is the hashtag of the j-th element of the hash chain of the verifier v, $@x$ is the reference to the user x, $\#TD_x^i$ is the hashtag of the ID of the *t-tweet* generated by x, and *status* can either be 1 for *success* or 0 for *failure* on the basis of the verification result. The remaining of this tweet is the essential part of the body of the *t-tweet* of x necessary to reconstruct the original tweet in case of deletion done by x.

Verification. This protocol is used to check the validity of a transaction. Now, suppose a verifier, say v, wants to verify the i-th transaction of the user x having the p-th transaction of a user y as input and the user r as target. The transaction of x will have the following structure: $\langle \#HC_x^i \#TID_y^p$ content $@r\langle$.

The protocol works by verifying each part of this *t-tweet*. Observe that, concerning the verification of the content, this is not considered here as it is strictly related to the objective of the transaction, which is not specified in this paper and, therefore, it is fully application dependent. As for the verification of $\#HC_x^i$, first v checks whether this hash chain element has been already used, in this case the verification will fail, otherwise the verifier has to compute the SHA-256 hash of $\#HC_x^i$. Due to the hash chain property, the results of this computation should be $\#HC_x^{i-1}$. Therefore, a search on Twitter for $\#HC_x^{i-1}$ should return the previous tweet posted by x. The goal of the verifier is to find the previous *t-tweet* of x or the initial `hello` message. Now, because x will also post confirmation

tweets for other users' transactions, in the case $\#HC_x^{i-1}$ refers to a confirmation tweet, the procedure above is repeated until either a *t-tweet* or the `hello` tweet is found. Let $\#TID_x^{i-1}$ be the ID of such a tweet, the verifier has now to check whether he has confirmed this tweet in the past (i.e., he has posted a confirmation tweet with status 1 corresponding to it). If this is not the case, then the verifier will not confirm (*status* 0) the new transaction, otherwise he will proceed with the verification of the second part of the new *t-tweet* of x. The verification of $\#TID_y^p$ implies the verification of the validity of the input transaction. As said above, the input is the p-th *t-tweet* of the user y. The validity of this tweet is related to the presence of at least t confirmation tweets among the s generated by the verifiers associated with y. Because each confirmation tweet contains the ID of the corresponding *t-tweet*, a search in Twitter for $\#TID_y^p$ will return all the confirmation tweets for the p-th *t-tweet* of p. Now, the verifier has to check both the presence of t confirmations with status 1 and that they have been posted by the legitimate verifiers for y. Moreover, v checks whether the target of $\#TID_y^p$ is x and that it has not been already used as input in any other transaction. The last verification done is the check of the existence of r (i.e., the target user of this new transaction) in the `Tweetchain` community. This is obtained by searching for @r in Twitter and by verifying the presence of the `welcome` message in W along with t confirmations of the verifiers of r. At end of these steps, if all the verifications succeeded, then the verifier will post a confirmation tweet with status 1 for the new *t-tweet* of x, otherwise the status of this confirmation will be set to 0.

4 Security Analysis

In this section, we describe a security analysis of our approach showing that it accomplishes its objectives also in presence of attacks. Therefore, in the following, we describe the security model and sketch the security properties of our proposal.

To analyze the security properties, first our threat model includes the following assumptions:

A1 The adversary cannot change information shown on the social network page of any user of the `Tweetchain` community.

A2 The Twitter service is up and running as expressed in its use conditions and does not intend to block the `Tweetchain` community (i.e., we assume that Twitter only could block single individuals, not the whole community).

A3 The cryptographic hash function is robust, in the sense that collision, preimage and second preimage attacks are infeasible.

A4 The adversary cannot access secrets information of users from which hash chains are computed.

A5 Among x verifiers, at most $m \cdot x$ out of them (with $0 \leq m < 1$) may not collaborate by executing correctly the *verification* protocol.

A6 At most t users can maliciously collude to break the security properties of the protocol.

Concerning the last assumption we have to recall that confirmations are generated collaboratively by multiple users playing the role of verifier. Some of these users might be corrupted by an attacker, but, as commonly assumed in the distributed domain [4,11], the majority of users at all times is considered honest. Therefore, our technique is parametric with respect to t. This value is chosen in such a way that the likelihood that at most t randomly selected users misbehave is negligible. Similar considerations can be done for Assumption A5, in which the parameter to set is m.

Now, we identify the security properties that our system has to assure and discuss the possible attacks.

SP1 - Transaction Authenticity. A transaction and the user generating it can always be verified by the `Tweetchain` community.

Attack AA1. *An attacker tries to impersonate the welcome profile W to tamper the list of verifiers for a user.*

Attack AA12. *An attacker creates multiple accounts and tries to use them as verifiers of his own transactions.*

SP2 - Transaction Integrity. The whole message (*t-tweet*) representing a transaction cannot be tampered once posted on the system.

Attack AI1. *Some of the verifiers do not execute the verification protocol invalidating a transaction.*

Attack AI2. *Some of the verifiers collude to compromise the integrity of a transaction.*

Attack AI3. *Twitter, playing as an attacker, tries to compromise the integrity of a transaction by deleting a portion of the chain (even the whole) this transaction depends on.*

SP3 - No repudiation of Transactions. The user generating a transaction cannot repudiate it.

Attack AR1. *An adversary tries to make ambiguous a transaction by forging another one with the same input transaction.*

Attack AR2. *The user, acting as an adversary, tries to repudiate a transaction by deleting the corresponding t-tweet.*

Being this paper a work-in-progress paper, we do not provide here the proof of the resistance of our protocol to the considered attacks.

5 Conclusion

Blockchain technology allows mutually distrustful parties to transact safely without trusted third parties and avoiding high legal and transactional costs. Despite the rapid grow of interest of both researchers and companies in Blockchain for all possible applications, Blockchain appears not suitable for small-value-transaction applications, typical of domains like crowdsourcing and WoT, due

to the payment of fees related to the proof of work. In this paper, we propose a lightweight public ledger that, instead of the P2P network and the protocol of Blockchain, leverages the popular social network Twitter, by building a meshed chain of tweets to ensure transaction security. Importantly, Twitter does not play neither the role of trusted third party nor the role of ledger provider. Moreover, no proof of work or fees are needed. Incidentally, the elimination of the proof of work results in another relevant advantage. Indeed, many miners (sometimes unsuspecting victims of an attack) compete to reach the goal but only one succeeds every time a block is approved. In other words, the protocol produces an extraordinary quantity of wasted work, and thus lost of energy. Therefore, the protocol is really little sustainable from a global energy consumption point of view.

As a future work we plan to formalize our protocol in a more abstract fashion, to deeply analyze it in terms of security, and to highlight its relevance in a real-life specific application setting.

Acknowledgements. This work has been partially supported by the Program "Programma Operativo Nazionale Ricerca e Competitività" 2007–2013, Distretto Tecnologico CyberSecurity funded by the Italian Ministry of Education, University and Research.

References

1. Bahack, L.: Theoretical bitcoin attacks with less than half of the computational power (draft) (2013). arXiv preprint: arXiv:1312.7013
2. Buccafurri, F., Lax, G., Nicolazzo, S., Nocera, A.: A model to support design and development of multiple-social-network applications. Inf. Sci. **331**, 99–119 (2016)
3. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: A system for extracting structural information from social network accounts. Softw. Pract. Exp. **45**(9), 1251–1275 (2015)
4. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Eur. Trans. Telecommun. **8**(5), 481–490 (1997)
5. Faye, Y., Niang, I., Noel, T.: A survey of access control schemes in wireless sensor networks. Proc. World Acad. Sci. Eng. Tech. **59**, 814–823 (2011)
6. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39884-1_2
7. Shultz, B.L., Bayer, D.: Certification of witness: mitigating blockchain fork attacks (2015)
8. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc., Sebastopol (2015)
9. Swanson, T.: Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems (2015)
10. Vasek, M., Thornton, M., Moore, T.: Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 57–71. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44774-1_5
11. Zwierko, A., Kotulski, Z.: A light-weight e-voting system with distributed trust. Electron. Notes Theor. Comput. Sci. **168**, 109–126 (2007)