Building a trusted ledger over Twitter

Francesco Buccafurri

DIIES Dept.,
University Mediterranea of Reggio Calabria, Italy
email: bucca@unirc.it

DLT workshop 2018, Perugia (Italy)

February 1, 2018

F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera, Overcoming Limits of Blockchain for IoT Applications. ARES 2017: 26:1-26:6

F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera, Tweetchain: An Alternative to Blockchain for Crowd-Based Applications. ICWE 2017: 386-393

Overview

Motivations

- **Blockchain** technology allows the implementation of a public ledger securely recording transactions among peers without the need of trusted third parties.
- A lot of challenging applications can benefit from the usage of a distributed shared trusted ledger.
- For some cases (eg., IoT and crowdsourcing applications) Blockchain technologies are not completely suitable.
- Indeed, it is not suitable for small-value transactions often occurring in the crowdsourcing paradigm.
- Low computational power and storage capabilities may limit the Blockchain use in IoT.
- In this talk we address this question: Can we implement a public ledger over a social network (e.g., Twitter) to overcome the above drawbacks?
- In this case, which is the price we have to pay?

An answer to the question: Tweetchain

- Tweetchain reinvents a consensus protocol using tweets to encode transactions and meshed replications (i.e., conformation tweets).
- Transaction and confirmation tweets from the community form a *meshed chain* making impossible for anyone, including Twitter itself, to alter, delete or forge valid transactions.
- All nodes participate in the same way, so that this protocol is truly decentralized.
- No P2P feature must be enabled on the client machine, but only an application working on the Twitter profile of the corresponding user.
- The role of Twitter is not that of a trusted third party. Not even a provider of some part of the protocol.
- This situation is very little plausible,

Overview of the Proposal: Tweetchain

- Transaction and confirmation tweets from the community form a meshed chain making impossible for anyone, including Twitter itself, to alter, delete or forge valid transactions.
- All nodes participate in the same way, so that this protocol is truly decentralized.
- No P2P feature must be enabled on the client machine, but only an application working on the Twitter profile of the corresponding user.
- The role of Twitter is not that of a trusted third party. Not even a provider of some part of the protocol.

Overview of the Proposal: Tweetchain

- Twitter could only deny the whole service, so stop the entire community. This situation is very little plausible. Twitter could be also be down for a while.
- In principle, multiple social networks can be used simultaneously.
- Tweetchain can be viewed as a *lightweight* public ledger for *non-critical services*, for
  - small-value transactions,
  - requiring fast validation (for Tweetchain is around 13 seconds),
  - no local storage,
  - no effort for low-computational-power endpoints (no cryptographic capabilities, no P2P download/upload traffic).

The Tweetchain Model

Actors of the model

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

Prerequisites

- To participate in the `Tweetchain` community, a user must be able to build, by starting from a secret, a (SHA-256 based) hash chain of size k (system parameter) which will be used to maintain all his timeline activities linked together.
- We assume that there are always at least $s = \frac{2t}{1-m}$ members in the `Tweetchain` community (m is the maximum fraction of users who do not collaborate and t is the maximum number of colluding users).

Registration

1. It is executed by each user, say x, who wants to become member of the `Tweetchain` community C.
2. The first step he performs is to follow the welcome profile W and to publish a `hello` tweet with the following structure:

$$\langle \#HC_x^1 \#HC_W^1 \text{ Hello } @W \rangle$$

   - 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
   - $@W$ is a Twitter reference to the welcome page W.

Registration

1. 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.
2. Suppose that W has already posted $i-1$ tweets, then the `welcome` message for x will have the following structure:

$$\langle \#HC_W^i \text{ Welcome } @x \#HC_x^1 \#TID_x^1 \rangle$$

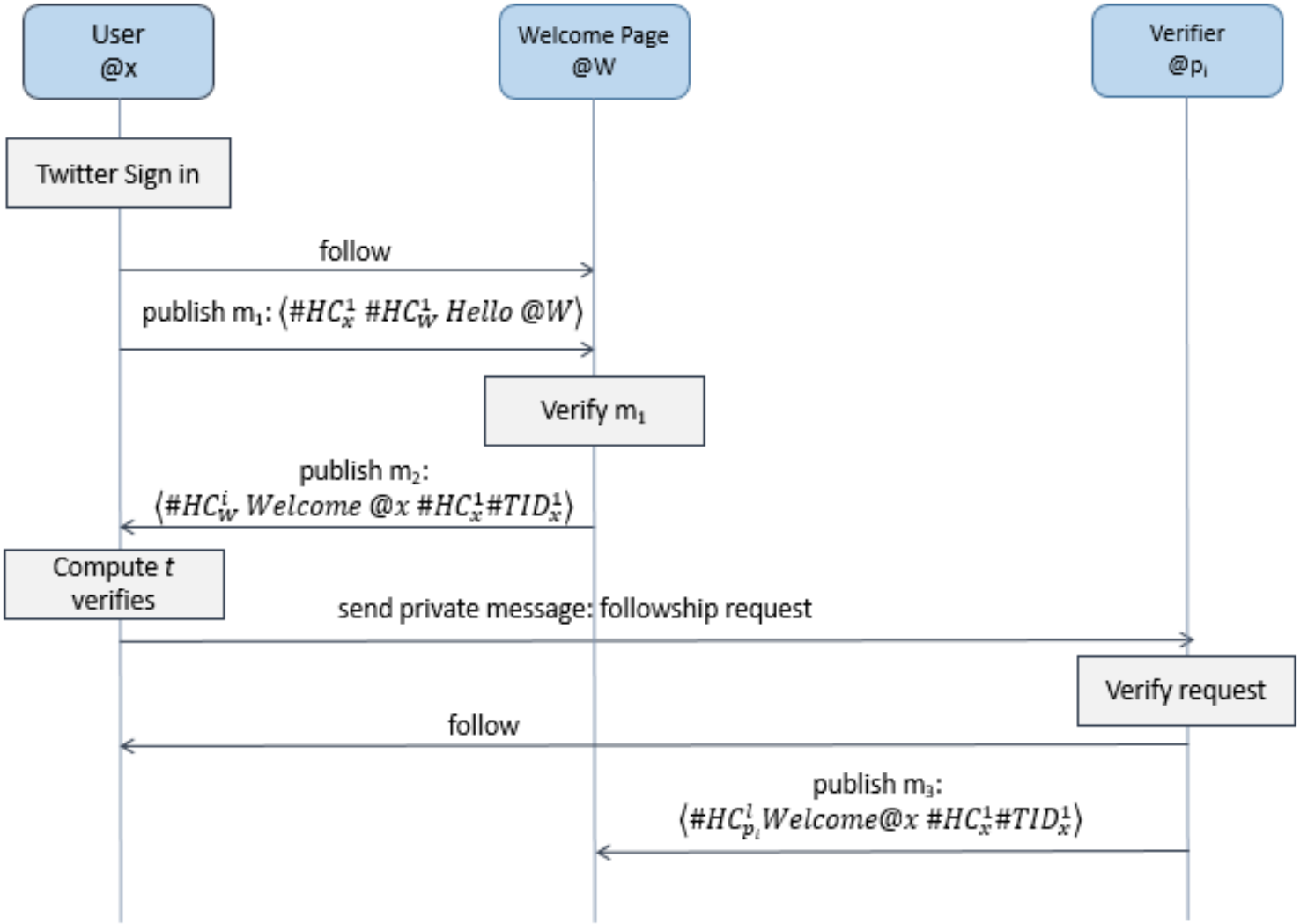
   - where $\#HC_W^i$ and $\#HC_x^1$ are Twitter hashtags of the base64 encoding of the ith element of the hash chain of W and the first element of the hash chain of x, respectively
   - $@x$ is a Twitter reference to the user x,
   - $\#TID_x^1$ is a Twitter hashtag with the ID of the first tweet (`hello`) of x as text.

Registration

After this, x generates at random the set F_x of followings who will validate his transactions in the future. This set F_x is built as follow:

- x retrieves his Twitter identifier.
- This identifier, concatenated with the position of the x's hello message in the W timeline, 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.
- 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 $\#HC^i_W$ with its current hash chain element.

Registration



Transaction generation

1. 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.
2. 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:
   $$\langle \#HC_x^i \#TID_y^p \text{ content } @r \rangle$$
   - 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,
   - $@r$ is a Twitter reference to the recipient r of this transaction.

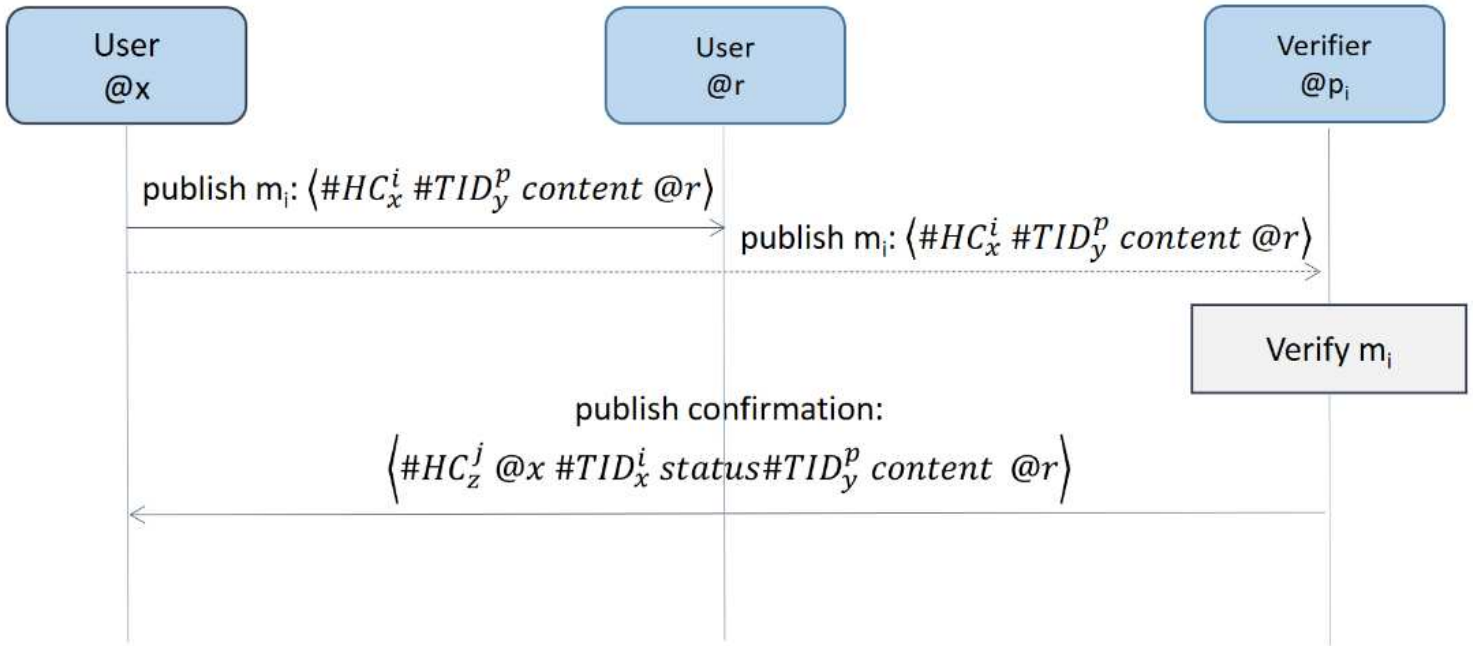
Transaction generation

1. 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.
2. They will proceed by verifying the legitimacy of this new transaction by using the verification protocol.
3. After running the verification procedure, they will publish a confirmation tweet on their timeline.

$$\langle \#HC_v^j @x \#TD_x^i \text{ status } \#TID_y^p \text{ content } @r \rangle$$

   - $\#HC_v^j$ 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
   - *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.

Transaction generation



Security analysis: Assumptions

1. The adversary cannot change information shown on the social network page of any user of the `Tweetchain` community.
2. 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).
3. The cryptographic hash function is robust, in the sense that collision, preimage and second preimage attacks are infeasible.
4. The adversary cannot access secrets information of users from which hash chains are computed.
5. 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.
6. At most t users can maliciously collude to break the security properties of the protocol.

Security analysis: security properties

1. **SP1- Transaction Authenticity.** A transaction and the user generating it can always be verified by the `Tweetchain` community.
2. **Attack AA1.** An attacker tries to impersonate the welcome profile W to tamper the list of verifiers for a user.
3. **Attack AA12** An attacker creates multiple accounts and tries to use them as verifiers of his own transactions.
4. **SP2 - Transaction Integrity.** The whole message (*t-tweet*) representing a transaction cannot be tampered once posted on the system.
5. **Attack AI1.** Some of the verifiers do not execute the verification protocol invalidating a transaction.

Security analysis: security properties

1. **Attack AI2.** Some of the verifiers collude to compromise the integrity of a transaction.
2. **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.
3. **SP3 - No repudiation of Transactions.** The user generating a transaction cannot repudiate it.
4. **Attack AR1.** An adversary tries to make ambiguous a transaction by forging another one with the same input transaction.
5. **Attack AR2.** The user, acting as an adversary, tries to repudiate a transaction by deleting the corresponding t-tweet.

Conclusion and Future Work

- Blockchain technology allows mutually distrustful parties to transact safely without trusted third parties and avoiding high legal and transactional costs.
- It appears not suitable for small-device and small-value-transaction applications, typical of IoT.
- 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.

Conclusion and Future Work

- In our protocol, Twitter does not play neither the role of trusted third party nor the role of ledger provider and no proof of work or fees are needed.
- The elimination of the proof of work makes our protocol more sustainable from a global energy consumption point of view.
- 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.

Thank You For Your Attention