Creating a Network for Identity and Key Attestations

Frankie Pangilinan Renée Pangilinan

November 2, 2022

Abstract

This paper presents a decentralized network protocol, based on Secure Scuttlebutt [TLMS19], for establishing identities through key attestations. In this paper we provide definitions of the components of the network topology with the intention that changes to the topology are reflected in the content of the network. We offer a method to recognize two separate single-writer append-only ledgers as a single individual.

1 Introduction

This paper offers a protocol for decentralizing identity and key attestations. Of high concern is the private and protected storage of personally identifiable information. Privacy implies that the user is the sole authority of their own data. Protection implies that the user has access to their data at all times. However, if a centralized server is down or corrupted, the user cannot access their identities, their login passwords, their friends, their messages and media, nor can they pick up this data and move it to another network. Accordingly, there is a strong push toward decentralizing these processes.

We will describe the elements of a decentralized network of trust. Particular attention will be paid to public messages and message types as a method of organizing data into unique identities. Finally, we will summarize the advantages of a network of trust, and procedures where an individual can recover their digital identity.

There are two levels of trust and identity concerning the network, both implicit and explicit. In each level, trust and identity are reflexive and symmetric binary relations of limited transitivity. To illustrate, Alice and Bob may trust each other, Bob and Charlie may trust each other, but Alice and Charlie may not necessarily share mutual trust. Alice has the option of extending trust to Charlie but it is not necessary. Implicit trust is external to the network and only measurable indirectly, through frequency of communication or some other statistic. Explicit trust and identity are internal to the network and are meant to reflect the implicit trust occurring offline. Explicit trust relies on granting arbitrary access to message metadata without necessarily allowing access

to encrypted message content so that there is consistency in the social graphs assembled. This notion of trust has two modes of verification: self-verification, where a person can prove their own identity; and group-verification, where a group can verify to each other the identity of a person. Digital identity emerges from well-maintained metadata. In this way, the group is secure against doppelgängers, and the individual is secure against conspiracy.

2 Network Topology

The Secure Scuttlebutt network is a mutable finite graph, not necessarily cyclic, where each node publishes messages which are propagated along the edges to other nodes. An edge may be added or removed from the network, thus altering the path of certain messages. Messages in the network may have a combination of certain qualities, ephemeral or persistent, unverified or verifiable, encrypted or public. Every node receives every message so that messages can be recovered throughout the network [TLMS19]. The network topology, at its coarsest, is a social graph where each node is an individual and each edge is a relationship between individuals. When two individuals, Alice and Bob, are introduced to each other, they can securely verify each other's identity and establish further communication with a secret handshake [Tar15]. This exchange adds to the graph an edge between the two individuals. The edge is reinforced with a public follow message from Alice and Bob. Alice will fetch and store Bob's messages, and vice versa, and the two can send private messages to each other. If Bob begins flooding the network with messages, Alice can, at any point, unfriend or block Bob, suspend all message syncing, and thus remove the edge between them. If enough of Bob's peers block him, then his messages will no longer be propagated through the network. However, once information has been revealed, there is no guarantee of deletion. Furthermore, if Alice was relying on Bob to store important information and Bob maliciously blocks her, she will still have her own local copy and can entrust it to someone else.

At a finer topology, each node is a device and interactions between individuals are facilitated by their devices. The devices are responsible for storing certain messages and fetching messages from other devices. This introduces a new type of edge into the graph since devices can be tied in a way that is stronger than an individual follow. At the finest level, each node in the network is a single-writer append-only ledger of well-formed messages, the author of which is represented as an ed25519 key pair, which they can use to verify the messages on the ledger [BDL+11]. Thus, the individual emerges as a collection of devices, each with its own collection of keys and ledgers, each with its own collection of messages; furthermore, each element must be attested and this attestation must be verifiable.

All information shared from one individual to another is in a message of some form. As a condition of trust, the information containing the topology of the network must be verifiable and publicly available to the network at all times. Since messages on the ledgers are immutable and self-certifying [TLMS19], they

are of the highest persistence and will be sufficient for defining a network of trust. An advantage of publishing the topology of the network is that when there are changes in one part of the network the rest of the network will be notified and can thus react appropriately.

3 Message Types

Given that the basis of the network topology consists of messages, this will be an apt place to begin with formal definitions. An author of these messages generates a private key and a public key, which are used to generate a verifiable signature over the message. Well-formed messages, along with a root message, are then organized into ledgers [TLMS19].

Definition 1 *A single-writer append-only ledger L is a set of well-formed messages $M = \{A, b, H, n, t, C, \sigma_A\}$. A is the author's public key, b is a backlink to the previous message of L, H is the cryptographic hash function used to compute b, n is the unique nonce assigned to M, t is the timestamp, C is the content, and σ_A is the author's signature over all the previous data.*

This organization of the messages ensures that the authorship and content of each message is verifiable. With this condition, it can be assumed that there exists some method of transmitting verified changes to the network topology. Either of two changes can occur: two nodes initiate a tie or one node cuts ties with another. To achieve this, the content of each message is given a type which provides information on the operation to be performed.

Definition 2 *The content C of a message M is well formed if $C = \{\tau, \delta\}$, where τ is the type and δ is the data.*

As a JavaScript object the message may look like this:

```
Message = {
    author,
    backlink,
    hash,
    nonce,
    timestamp,
    content,
    signature
}
```

The content itself may be arbitrary, or it can also be organized into key-value pairs.

```
{content: {
    key_1: value_1,
    key_2: value_2,
        .
```

```
        .
        .
    key_n: value_n
    }
}
```

The keys of 'type' and 'data' are especially useful in the process of organization. The protocol here established will recognize two new values for message type: 'tie' and 'cut'. A valid 'tie' between two ledgers requires an initiating key attestation from one ledger and a verification from the other. The first message can point to the public key being attested to, and the second message can point back to the original message.

Type 3.1 (tie) *Let $\{\tau_1, \delta_1\}$ be the content of message M_1 written by the author with public key A_1 and let $\{\tau_2, \delta_2\}$ be the content of message M_2 written by the author with public key A_2. If $\delta_1 = A_2$, $\delta_2 = M_1$, and $\tau_1 = \tau_2 =$ 'tie', then M_1 and M_2 form a verifiable 'tie'.*

The 'tie' messages induce an equality relation on the network. For any three ledgers x, y, and z: x is tied to itself; if x is tied to y then y is tied to x; and if x is tied to y and y is tied to z then x is tied to z. In implementation, transitivity of the 'tie' can have a maximum depth, contained as a key-value pair in the content of the message, in order to minimize the transmission of irrelevant information. For example, if ledger x is tied to y ad y is tied to z, a 'tie' with a depth of 1 between x and y will potentially extend the 'tie' from ledger x to ledger z. However, if z has restricted its maximum 'tie' depth to 0, x will not be able to indirectly tie with z.

After two ledgers have been tied either ledger can cut ties with the other. The 'cut' message must be symmetric only. Ledger x can not cut ties with itself, so 'cut' is non-reflexive. Additionally, 'cut' is non-transitive. If a group of ledgers are tied and the private key to ledger z is compromised or due for regular maintenance, the other ledgers can cut ties with z without cutting ties with each other. A valid 'cut' must come from one of the two tied ledgers and point to either of the previous 'tie'-type messages.

Type 3.2 (cut) *Given that messages $M_1 \in L_1$ and $M_2 \in L_2$ form a verifiable tie, if there exists a third message M_3 such that $\tau_3 =$ 'cut' and $\delta_3 = M_1$ or $\delta_3 = M_2$, then M_3 forms a verifiable cut.*

When Alice syncs her feed and receives her peer's messages, storing her own copy of them, each ledger is assembled and scanned for 'tie' and 'cut' type messages. Each 'tie' message will either link out to another ledger or link back to a previous 'tie' message. Each 'cut' message always links back to the previous 'tie' message on the same ledger and terminates the relationship.

4 Identity

This notion of a sovereign individual that owns and can alter a subset of the network at will deserves some finer definition. It is not unheard of for an individual to require many forms of identification from a sovereign government or a centralized network, who own and manage the separate identities. Yet the identity is not the sovereign individual who may act in defiance of any official identification. For the decentralized government of the network, however, each individual owns and manages their own subset of the whole graph as well as their own identities. Identity within the network takes the form of tied ledgers.

Definition 3 *An identity $\mathcal{I}$ is a collection of ledgers L such that for each L_i and L_j, where $i \neq j$, there exist messages $M_i \in L_i$ and $M_j \in L_j$ that form a verifiable tie.*

Provided that the private key of a ledger is only stored in one location, that is one device, then knowledge of the private key implies ownership of the ledger, and thus ownership of the identity tied to the ledger. These key attestations allow for easier management of assets and securities. When a peer or other third party requests information, the request can be made to any one of the attested keys in a legitimate tie. In the network of trust, the individual acquires sovereignty of their data and which data other individuals can access.

5 Recovery

An advantage to a network of trust is having a circle of trusted peers who can verify each other's identity. In the event that Alice's digital identity is compromised, one of her private keys is lost or stolen or a fork appears on one of her ledgers, she can rely on her peers to aid in the recovery process.

Although each ledger is append-only, there may be a fork in the feed; that is, two messages, otherwise well-formed and verifiable, have the same nonce. The protocol will reject new messages and cap the ledger from further updates [TLMS19]. If the fork can be resolved, however, then a main identity can be recovered. Provided that each private key is only stored on a single device, then a fork can only occur if a private key has been stolen or misdelegated. Since the entire network will see the fork, they can notify Alice and request a next action. We propose a 'cork' type message, verified by Alice's peers and appended onto the invalid message. Thus, the whole network will cork the fork and allow the original branch to continue.

Another concern is private key and seed phrase recovery. When Alice generates a private key from a seed phrase, she has the option of setting up a recovery process with a secret sharing scheme. The classic (t, n) scheme, where n is the number of distributed shares and t is the threshold, assigns the secret as the constant term of a polynomial of degree $t - 1$ and each share is a point on this polynomial. Then, any t points can reconstruct the secret [Sha79]. Alice can designate shares of her seed phrase to some of her trusted peers, encrypted as

an extra security measure. If a device is lost, stolen, or destroyed, Alice can request the shares back from her friends and recover the compromised keys. This is especially useful for inheritance, should Alice die and her heirs need to recover her digital identity. Otherwise, the risk of conspiracy can be avoided with several obfuscations. For one, Alice's friends do not necessarily know who has the other shares nor how many are required for the threshold. Secondly, Alice can induce a secret ordering of the shares. Since each share is a point on a polynomial, the $x-$coordinates determine an ordering of the shares, the value of which can be withheld from the share holders. Alternatively, Alice can withhold the last share and split this share again. This produces several layers of shares with different priorities in reconstructing the secret.

6 Conclusion

This paper scrutinized the network topology outlined by the Secure Scuttlebutt protocol. With the insight that the movement of messages is the basis for the shape of the network, we proposed a method for using message types to recognize individuals and their digital identities within the content of the messages. The message type 'tie' is used for binding two ledgers into a single identity and the message type 'cut' is used to unbind the ledgers. The public declarations of the topology of the network increase implicit trust between the peers in the network. With a decentralized network of trust, the management, recoverability, and sovereignty of an identity are returned to the individual.

References

[BDL+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang4. High-speed high-security signatures. 2011.

[Sha79] Adi Shamir. How to share a secret. 1979.

[Tar15] Dominic Tarr. Designing a secret handshake: Authenticated key exchange as a capability system. 2015.

[TLMS19] Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Schudin. Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications. 6th ACM Conference on Information-Centric Networking (ICN '19), September 24–26, 2019, Macao, China. ACM, New York, NY, USA, 2019.