Marc Ruiz Rodríguez

TallyNetworks

Protecting Your Private Opinions with Edge-centric Computing

Master's degree final project

Directed by Dr. Pedro García López

Master's Degree in Computer Security Engineering and Artificial Intelligence

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2017

"Without privacy there was no point in being an individual"
— Jonathan Franze, The Correction

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Pedro García for the useful comments, remarks and engagement through the development process of this Master thesis. Furthermore, I would like to thank Dr.Pedro García and Dr. Marc Sànchez for introducing me to the topic as well for the support on the way and their priceless contributions to the final result and their positive and constructive attitude during all the evolution and reviewing of the project.

Also, I would like to thank the European Union and the Spanish Ministry of Science and Innovation as work has been partly funded by the EU project IOStack (H2020-644182) and Spanish research project Cloud Services and Community Clouds (TIN2013-47245-C2-2-R).

Finally, I would like to thank my loved ones, who have supported me throughout the entire process. I will be grateful forever for your love and support Enric, Yolanda and Sílvia.

Abstract

In this work we claim that your private opinions cannot be controlled by a single centralized entity. In this direction, we present TallyNetworks, an edge-centric distributed overlay that aims to provide end-to-end verifiability of online opinions by leveraging the computing resources (TallyBoxes) of users and third-party organizations. This will (i) securely compute votes/opinions in a large group, (ii) guarantee privacy, integrity, robustness and verifiability, and (iii) ensure that opinions cannot be tampered with or censored in any way by third parties. Through blind signatures, pseudonyms and anonymous channels, it is possible to ensure that the edge nodes (TallyBoxes) are blind and guarantee anonymity and privacy. Thanks to a one-hop structured overlay and a global membership protocol using redundant broadcasting and syncing, we guarantee that messages reach all nodes in the network (integrity, robustness), and that vote information can be obtained and checked from different points (end-to-end verifiability). Some examples of this are user participation in open polls or rating (`stars`, `like`/`dislike`) services and persons in a community.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Nowadays, people like to check beforehand how good the restaurant they will go next weekend is or how suitable the item they want to buy is in their favorite on-line store. Not only that but they also want to know what polls predict about the next election winner or what people think about the last news we have seen in our social network (they like, love, hate... them). Mainly, we can say that today people like to know the public opinion (ratings, polls, like/dislike...).

However, on these days it seems that we live in a post-privacy world where our opinions are controlled by a few big players in the market. Namely, the business models of companies such as Facebook or Google rely on analyzing user opinions and behaviors and trade this valuable information with advertisers.

Furthermore, a flourishing market of data brokers is emerging to analyze consumer's data to create consumer's profiles which may contain sensitive user information. In fact, every time you like a post in Facebook, whenever you like or dislike something in Google or Youtube, when one rates something in Amazon, or anytime you participate in a poll, you are giving away valuable information about yourself.

Although the fact that when we rate or like something in an on-line store or a social network it may seem an innocuous action that sometimes can even be funny, being able to gather all your ratings, likes and opinions, allows such big players to process this information to build your on-line identity. Therefore, this can finally have an affect on your reputation. In fact, it can even lead to you paying an increased price for some services such as health or driving insurances due to some acts reflecting bad reputation or personal public image in specific areas.

Additionally, we have to consider that such big players use all such information to tell your friends, neighbors or people you may know, that you liked something to recommend it to them. This actually means that they are using you to sell products. Moreover, they are

telling to others what things you liked or rated and if you are not very careful on what you like or rate, it may reveal private preferences that you would not want to be revealed or being controlled by a third party.

Following the claim of edge-centric computing [18], we believe that key personal and social communication services should be decentralized and human-driven. In this Master thesis, we introduce TallyNetworks, an edge-centric distributed architecture designed to preserve your opinions/voters' privacy in large on-line communities in which public opinion matters, but not the identity of each of the persons that expressed their opinion. At the same time, we empower people again by centering the potential on the edges of the network, that is, returning them the control of their own information instead of trusting and giving it away to a third party.

The core idea of our system is to move opinion counting and storage to the edges of the network which are powered by the users. A key assumption is that we cannot trust a single centralized entity to store and count our opinions because of the above mentioned reasons. Instead, we will rely on a decentralized and privacy-oriented overlay of TallyBoxes which are controlled by the users of the system.

Our model is inspired in the remote voting paradigm where participants receive blindly signed voting credentials that permit them to vote anonymously. However, there is a key difference with classical voting systems in which the result is only given at the end of the election. The contracts is that in our case, you must be able to provide real time results. Like remote voting systems, we provide:

(i) *Privacy:* the identity of a voter cannot be linked to his vote.

(ii) *Integrity:* the result of the election cannot be altered in any way.

(iii) *Robustness:* the tolerance to the misbehavior of users or external parties.

(iv) *End-to-end verifiability:* the check that the reception and tallying of the votes is correct.

The key insight behind our approach is to leverage the services provided by a decentralized, one-hop overlay of Tallyboxes and combine them with cryptographic techniques such as blind signatures, pseudonyms and anonymous channels. The stability and capacity of edge nodes (that are mainly stable home devices, datacenter edge services and nano datacenters) permits the management of a global routing table and the one-hop abstraction that we will use to distribute and count all the votes and opinions of the network.

As a proof-of-concept of TallyNetworks, we implemented a global membership service based on a robust and reliable broadcast and synchronization algorithm for Kademlia, which is

a well-known peer-to-peer distributed hash table, and we used it to simulate the broadcasting of votes and opinions with different levels of malicious nodes. Notice that our algorithm permits to set the targeted robustness to attacks and churn by tuning redundancy across both the defined phases or steps: broadcast and synchronization. Furthermore, we can dynamically adapt such parameters in order to evolve the network with the circumstances.

This work was presented in the 22nd International European on Parallel and Distributed Computing Conference (Euro-par)[1], specifically in the 4th Workshop on Large Scale Distributed Virtual Environments (LSDVE)[2] and later published [37] as our contribution with the aim of moving and pushing the world from a post-privacy to a privacy-first era.

[1]For further information, visit the conference website: https://europar2016.inria.fr

[2]For further information, visit the workshop website: http://pages.di.unipi.it/ricci/LSDVE16/LSDVE16.html

Chapter 2

Background

The TallyNetworks model is an edge-centric distributed architecture that is inspired in the remote voting paradigm where participants receive blindly signed voting credentials that permit them to vote anonymously. However, anonymity must also be guaranteed during the communication phases in order to avoid linking the pseudonymous voting credential with the one that emitted the vote. This is why anonymous communication channels will also be necessary in our proposal.

Therefore, in this chapter, edge-centric computing, blind signatures and anonymous communication channels are described in detail as they form the main background for our proposed model. Moreover, DHTs (Distributed Hash Tables) and specifically Kademlia, are also explained since our proposed method will base its global membership protocol and topics distribution on it. Those will ensure scalability and a reliable information management in a distributed environment.

2.1 Towards a decentralized shift

Throughout the human evolution and in a wide range of human activities, the struggle between centralization and decentralization has been emphasized. Some examples are the power shifting in federal states, in which more or less power is given to the central state; or the conversion of energy generation in decentralized grids, when it was first concentrated in large power plants, to name but a few. The history of computing also shows this changes between centralized and decentralized control. For instance, in the 1980s, due to the Internet emergence, the systems gradually moved from centralized main-frames to PCs and local networks, and terminated in fully decentralized systems using peer-to-peer performing autonomously in a distributed fashion.

However, some issues arise in the decentralized approach that lead to the need of a new shift. Firstly, not only the number but also the diversity of powerful computing devices at the user-facing end of the Internet has significantly increased. This is clearly shown by the boost of high capacity mobile devices, which are always on; or home routers and dedicated Internet connection boxes. Another example is the increase of high-bandwidth pervasive wireless networks. Secondly, there has been a growth in the users concerns about trust, privacy and autonomy given by cloud services, and this further accentuates the decentralized approach.

Therefore, these problems prompt the action of taking the control of computing applications, data, and services away from some central nodes (the "core") to the other logical extreme (the "edge") of the Internet. This new shift in the history leads to the Edge-centric Computing.

2.1.1 Drawbacks of full centralization

It is widely known that cloud computing has an enormous capacity to create effective economies of scale using its dedicated data centers and simple centralized architectures. With cloud computing, users can easily scale services to fit their needs, customize applications and access cloud services from anywhere with an Internet connection. Enterprise users can get applications to market quickly without worrying about underlying infrastructure costs or maintenance.

However, as stated before, full centralization has significant drawbacks [18] when pushed to such a logical extreme:

- **Loss of privacy**: Centralized service providers have amassed unprecedented amounts of data about the behaviors and personalities of individuals. This is a fundamental problem given by the fact of releasing personal and social data to centralized services. Some examples are e-commerce sites, rating services, search engines, social networks, and location services.

- **Complete delegation**: Another fundamental problem is the complete delegation of the applications and systems control from the users to the cloud. Organizations are encouraged to use Cloud Computing as they provide a solid environment for planning the needed resources, cutting the business costs and applying the safety standards on the highest level [11]. However, this requires unilateral trust from clients to the clouds as they entirely depend on them. Moreover, this prevents from establishing finer grain trust between users.

- **Non-exploitation of personal devices**: The growth of the power of modern personal devices is not exploited when all the work is done in centralized cloud computing services. Therefore, there is a missed opportunity to exploit the huge amount of computational, communication, and storage capacity of those devices.
- **Absence of human-centered designs**: Centralization also prevent the creation of novel designs centered in humans, that may lead to the emergence of novel applications and the strength of the human-machine interaction.

2.1.2 Edge-centric Computing

All these disadvantages of cloud computing described in Section 2.1.1 yield the propose of another shift to a new decentralization: the edge-centric computing.

Edge-centric computing has been recently proposed as the natural evolution of peer-to-peer (P2P) systems. One of the major claims of this new paradigm is to move the control to the edges of the system (human-controlled) in order to preserve user's privacy. Edge-centric systems may combine centralized and decentralized components in order to overcome the limitations of P2P systems.

Applying this paradigm will lead a movement of the data and computation from the centralized nodes to the edges of the network.

This paradigm has a double objective:

1. Although we want to make a shift, the new paradigm needs to take advantage of the main interesting properties of clouds. To do so, they will be integrated as a support infrastructure. We position that the power of cloud computing needs to be exploited as well in this new shift, but in a different form.
2. The creation of new human-centered computing applications wants to be encouraged by returning the control and trust decisions towards the edges.

In Figure 2.1 it can be seen the structure of an edge-centric computing model. From the illustration, three layers can be differentiated, which are the important components in describing this new paradigm:

1. **Core**: It is the most inner layer and consists of the clouds and data centers. As mentioned before, in this new paradigm we want to include clouds services as well, although in a different way.
2. **Intermediate layer**: It encloses the core and this is where there are smaller servers and content distribution.

3. **Edge**: It surrounds the intermediate layer and consists of the user's devices such as desktop PCs, tablets and smart phones. A part from that, we can also find in here nano data centers, which are stable computing devices such as routers or media centers.

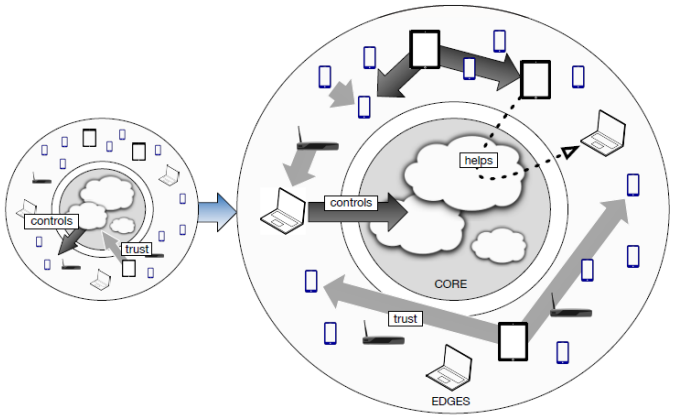


Fig. 2.1 Centralized cloud model (left) versus Edge-centric Computing (right) [18].

The main characteristics of edge-centric computing are the following:

- **Use of the proximity of the edge nodes**. In this new model there exists the sense of proximity between neighbor nodes. As it is known from P2P systems and content distribution networks (CDNs), the exchange of information and also its distribution is always more efficiently carried out when the nodes are close to each other. Edge-centric computing can benefit from it in contrast of using perhaps-unnecessary centralized intermediates that are further away between nodes in the edge.

- **Exploitation of powerful edge devices**. As technology evolves, the devices used for humans are increasingly becoming more intelligent, with more capacity and with even a smaller size. This yields the opportunity to perform autonomous decision-making in the edge. Some examples are novel distributed crowdsensing frameworks [17], human-controlled actuators [26] and agents reacting to the incoming information flows [32].

- **Increase of the user's trust**. The new paradigm allows the data of the users to be pushed to the edges. Therefore, personal and social data relative to the user is stored in her control. The management of the sensitive data is carried to the edges, which may reinforce a secure and private management. This all leads to an increase of the user's trust towards the system.

- **Control is in the edge**. Another property is that the control of the applications is also done in the edges. The edges are deciding where the data is stored and how it is distributed and synchronized. Also, they choose when they delegate some storage or computation to other nodes or to the core. Therefore, we can see that in this new paradigm the devices of the users are the coordinators of the activity of the applications.

- **Human-oriented design**. By applying edge-centric computing, we are encouraging the creations of new applications which are crowdsourced. This means that data of different edge devices is aggregated and correlated to form the application. This clearly leads in users having more control of the links of their networks. In general, we believe that with this shift we are giving the opportunity to create applications in which the power is in the crowd of users.

2.2 Anonymization Techniques

As we previously stated during the introduction, we aim to build a system that preserves user's privacy, but also their anonymity when participating in the network. For this reason, we need to use anonymization techniques in two different situations. However, first we will briefly introduce the terminology related with anonymity that can be found in [31] and [10].

- **Anonymity:** This concept is fairly intuitive. It is defined as *the state of being not identifiable within a set of subjects, which is called the anonymity set*. We can define it in a more formal way saying that a subject is anonymous if the probability of identifying her in the anonymity set, which includes all the possible n subjects, it is exactly $\frac{1}{n}$.

- **Unlinkablity:** It is formally defined in the [ISO15408 1999]. In its definition we can distinguish two kinds of unlinkability: absolute unlinkability and relative unlinkability.

  The first one refers to the fact of not being able to determine a link between uses of a resource, i.e., we cannot determine if a single user used multiple times the same resource or if such resource was used by multiple users; while the second refers to the fact that the knowledge about the system does not increase by its observation by an attacker. In other words, the attacker won't learn about links between resources by the observation of the system.

- **Unobservability:** Its formal definition would be: *It is the state of items of interest (IOIs) being indistinguishable from any IOI (of the same type) at all.* Roughly speaking,

it means that, if there is sender unobservability, nobody will notice if the user sends a message. On the other hand, if there is receiver unobservability, it means that nobody will notice if the user receives a message. Therefore, with both combined, nobody will notice that a message has been exchanged between two users, so a message will be indistinguishable from a random noise.

- **Pseudonymity:** A pseudonym can be defined as an identifier of a subject or set of subjects that do not change during the time unless their pseudonym is transferable. Therefore, in pseudonymity, instead of using their real identity, they use a pseudonym.

As stated before, in the case of our proposal, there are two situations in which we will need anonymization techniques. On the one hand, we need such techniques in order to protect the real identity of the person when participating in the network. However, anonymizing her would not allow the system to prevent her from participating more than once in the same election. For this reason, we will use pseudonyms that can be obtained through blind signatures, inspired by previous works on remote voting that will be briefly described in related work section.

On the other hand, we will need anonymization techniques for some of the communications in order to ensure that devices, IPs or other identifiers subject to communications cannot be tracked or linked to the pseudonym that the real person is using to protect her identity. Therefore, we ensure that the real identity remains anonymous.

Combining both anonymization channels and pseudonyms, we can reach a level of pseudoanonimity that is by far sufficient for the problem of distributed management of personal opinions as it will be shown during the explanation of our proposal.

In the following subsections we will introduce the main ideas and background behind the above mentioned techniques.

2.2.1 Blind Signatures

Although blind signatures [4] have a more general purpose than being only used for anonymization purposes, we decided to include them in this anonymization section because, as you will see in our described proposal in chapter 4, we use them in order to create a pseudonym for the users of our network in order to make them pseudoanonymous.

A blind signature allows a user to get a signature on a hidden message without the signer learning the message in question. Therefore, in this form of digital signature, the content of a message is disguised (blinded) before it is signed. Then, when the message in unblinded with the original message, the blind signature can be publicly verified against that, just as a regular digital signature works[45].

Due to its characteristics, blind signatures are usually used in protocols related to privacy. These protocols are needed when the signer and the message author are different entities. Some examples of applications, which in fact will be used in TallyNetworks, are cryptographic election systems and digital cash schemes.

A secure blind signature scheme ensures two characteristics:

- **Unforgettably**: this property ensures that nobody can fake a new signature for a new message.
- **Blindness**: the signer will never be able to learn the message she is signing nor be able to link a signature to the protocol run where it was obtained.

We can define the analogy illustrated in Figure 2.2 in order to make things clearer. Consider Alice has a letter which should be signed by an authority, Bob. However, Alice does not want Bob to be able to read the letter. Thus, she finds a clever solution: she can place the letter in an envelope lined with carbon paper and ask Bob to sign it. Bob will sign the outside of the carbon envelope without being able to open it, so when he sends it back to Alice, she will be able to open the envelope and find the letter signed by Bob, without him seeing the contents.

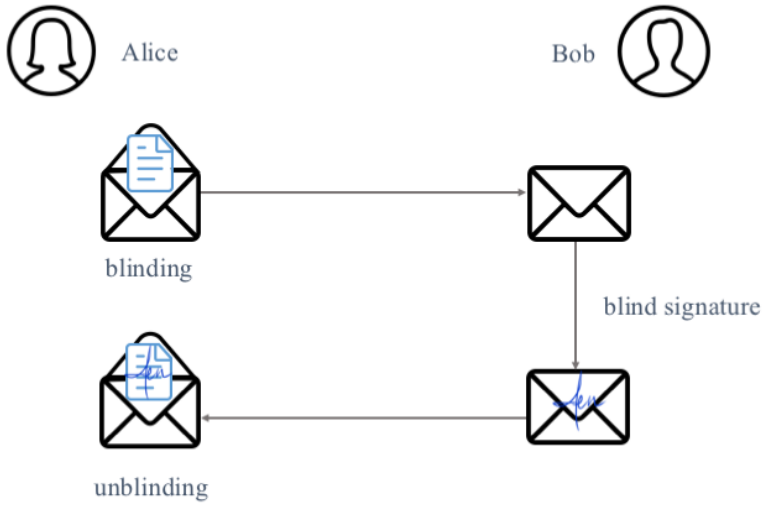


Fig. 2.2 Blind signature diagram

Two operations can be distinguished:

- **Blinding** $\mathscr{B}(m)$: the process of putting the letter m into the envelope to avoid Bob seeing the content.

- **Unblinding** $\mathscr{U}\mathscr{B}(m)$: the process of taking out the signed letter m from the envelope.

Following the previous examples the operation steps would be the following:

1. Alice "blinds" the message m, with some random number b (the blinding factor). This results in $B(m,b)$.
2. Bob signs this message, resulting in $sign(\mathscr{B}(m),d)$, where d is Bob's private key.
3. Alice the unblinds the message using b, resulting in $\mathscr{U}\mathscr{B}(sign(\mathscr{B}(m,b),d),b)$.
4. The functions are designed so that this reduces to $sign(m,d)$, which is Bob's signature on m.

2.2.2 Anonymous communication channels

Taking into account our proposal, although the use of blindly signed pseudonyms should guarantee that the real identity of the participant is never revealed, such pseudonym could be linked to the real identity of the person through the connections that the user establishes with other nodes in the network.

In order to achieve unlinkability between the real identity and the pseudonym through the established connections, we will need an anonymous channel which guarantees that such link can never be established.

In the field of anonymous channels there is a lot of literature and previous work. This is because behind the idea of anonymous channels there are a lot of kind purposes and of humanity interest such as freedom of speech, avoid surveillance or data retention, access to censored content,...

In the cited survey of anonymous communication channels [10], we can find most of the work around anonymous channels. However, in this thesis, as the anonymous channel will only be a means, and not the main focus of study, we will briefly introduce some of the proposed solutions in the literature in order to understand the main ideas behind anonymous communication channels. Finally, we will study with a bit more of detail one of the most popular and most studied solutions according to [31].

One of the first proposed solutions in the literature is using trusted or semi-trusted relays. Those are systems that introduce a central entity in which users rely to provide security for them. However, this option normally provides a low degree of anonymity protection against traffic analysis and active attacks as it would be easy for an attacker to trace the user if the central server is known or even pretending being such server. A current offered solution

based on this approach is Anonymizer [1] which provides a kind of proxy server to ensure the privacy of the users. However, this anonymous scheme is too weak for our purposes as previously described.

Another popular solution based on trusted servers to rely on is Crowds. It basically works as follows: each user contacts a central server and receives the list of participants, i.e. the "crowd". A user then relays her web requests by passing it to another randomly selected node in the crowd. Once the node receives the request, it randomly decides if it should relay it further through the crowd or send it to the final recipient. Finally, the reply is sent back to the user via the route established as the request was being forwarded through the crowd. However, it has been demonstrated, as stated by [10], that some attacks are feasible to discover the identity and the content of the messages since they do not use encryption between servers. Therefore, this system also offers a weak solution for our purposes.

Besides trusted or semi-trusted relays solutions there are other kind of solutions. Among them, the most noteworthy are those based on Mix Networks. The general solution for such networks, as defined in [44], consists on using a chain of proxy servers, also known as mixes, that take in messages from multiple senders, shuffle them, and send them back out in random order to the next destination. The receiver can be another mix node or the final destination depending on the length of the chain.

Furthermore, messages are encapsulated in layers of public key encryption in which each layer corresponds to the encryption for a certain proxy. Each proxy server strips off its own layer of encryption to reveal where to send the message next.

This concept was first introduced by Chaum [6] and nowadays it is one of the most popular and secure solutions based on mix-networks. Also, one of the most studied is Onion Routing and specifically its second generation version, TOR, as stated in [31].

Based on the literature, Onion routing and specifically TOR seem to be some of the most noteworthy solutions in terms of anonymity and security. Therefore, we decided to use it as our anonymous communication channel and in the following lines we will describe how it works with more detail.

TOR: The new generation onion router

The paper [13] describes TOR as the second generation of onion routing. The main idea behind it, the same as mix networks, is encapsulating a message in layers of encryption analogous to layers of an onion as it has been depicted in figure 2.3 for a better understanding and easier visualization.

[1] For further details about Anonymizer visit: http://www.anonymizer.com/

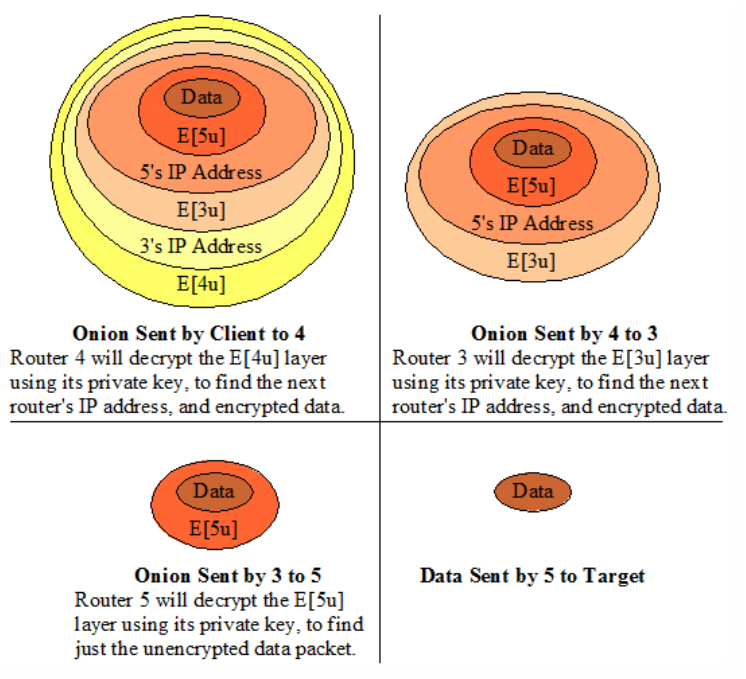


Fig. 2.3 Onion layers depicted with an example [27]

Each layer of encryption corresponds to a proxy in the path to our destination. The construction of this path starts when Alice uses her TOR client to obtain a list of TOR nodes from a directory server with the aim of accessing any service through the anonymization network. Once she receives such directory, she will pick a random path to the destination server. The connections between each proxy are encrypted with TLS except the final destination (the website or service we want to access) that will be only encrypted if it supports or offers such feature.

Alice will encapsulate the message in encryption layers, one per each hop (proxy) in the path after negotiating the keys as shown in the protocol of figure 2.4. Each proxy will then strip off one layer of encryption so he will be able to see the next destination. Once the next destination is obtained, he will forward the message until it reaches the real final destination.

Once Alice want to visit a different service or website, the process will start again with a new random path.

TOR is also known as the second generation onion routing as besides all the properties provided in the first onion routing definition, it adds perfect forward secrecy negotiating temporal keys and using different random paths for accessing resources. TOR also adds congestion control, directory servers, integrity checking and configurable exit policies. Furthermore, it adds telescoping circuits, which besides helping to circumventing the original patent of Onion Routing, it is the responsible of providing the above mentioned forward

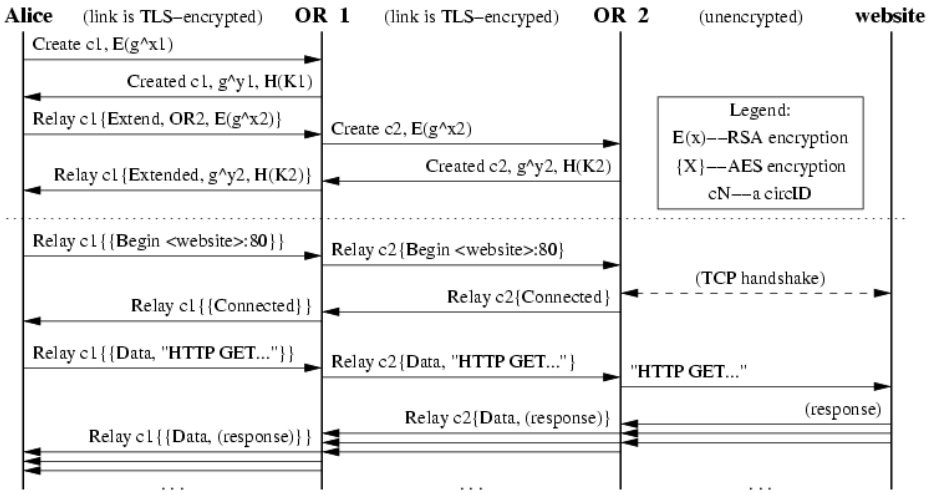


Fig. 2.4 Onion layers depicted with an example [13]

privacy. The original Onion Routing proposal used long term keys, but telescoping circuits consists on negotiating short-term session keys with each node along the path which will be then used to encrypt the onion layers of the message.

Thanks to the anonymization provided and the strong security that TOR provides, it makes TOR a perfect tool for our anonymization purposes.

2.3 Distributed Hash Table

A key assumption in our proposal is that trusting a central entity to store and count our opinions leaves people in a vulnerable situation that may compromise their privacy. Therefore, we claim that information should be distributed and human-driven.

For the above reason, we require an scalable but also reliable mechanism to distribute opinions and maintain the membership of the network. In this sense, DHTs are known to be an excellent solution for distributing information and looking it up in an scalable and reliable way which perfectly fit with our requirements.

The main objective of hash tables is the insertion and retrieval of key-value pairs in an efficient way and therefore a Distributed Hash Table (DHT) [25] aims to provide the same functionality but in a decentralized way. In a DHT, the hash table is distributed among the nodes so that each node in the system stores a part of it.

In order to provide an efficient lookup and insertion of the keys despite the distribution of the table over the network, the nodes are interconnected in a structured network overlaying the system. Thus, a user requesting a key in a node which does not contain the corresponding

hash table part still is guaranteed efficiency in finding the key in the other nodes due to this structured connection between them.

Moreover, as nodes can join or leave the system, the overlaid network is maintained and replication of key-value pairs is applied to multiple nodes. This is done to guarantee robustness.

Therefore, the structure of a DHT is basically composed by [47]:

- *Keyspace*: It is the foundation of a DHT and it defines the range of the keys. An example could be the set of 160-bit strings.
- *Keyspace partitioning*: it is a scheme that splits the keyspace among the nodes so that each node owns a subset of the keyspace. In other words, it is the map from key to nodes. There are multiple variants to make this map function but the essential property that has to provide is that when a node is removed or added in the system, only the nodes with adjacent IDs will have to change their set of keys, leaving all the other nodes with the same set of keys they had. Recall that in hash tables usually the addition or removal of one bucket implies the redistribution of almost all the keyspace. In a DHT this would not be efficient, because this would mean an intensive data movement through the bandwidth, given by the move of objects stored in the DHT. For this reason, the property explained before is important to keep the reorganization minimal when a node arrives or fails.
- *Overlay network*: As explained before an overlaid network is required to connect the nodes so that any node can find a given key in the keyspace. The way in which this network works is by each node maintaining a set of links to other nodes. This is called a routing table, where all the routing tables from each node form the overlay network. In order for a node to choose the set of neighbor nodes (nodes in the routing table), a certain structure is followed that is called the network's topology. As in the previous point, here there are multiple variants that have an essential property: for any key in the keyspace, each node in the system either has a node ID that owns the key or it has a link to a node whose node ID is close[2] to the key.

Although there are a lot of different protocols or implementations of DHTs such as Pastry [38], Chord [42] or P-Grid [1], we focused on Kademlia because of its unique combination of features that will be described in the following section and that will be really useful in the implementation of our proposal.

[2]For some distance definition, such as geographical or network latency.

2.3.1 Kademlia

Before pointing out the benefits that Kademlia can achieve to help us with our proposal implementation, we will first introduce how it works and its main characteristics.

Kademlia [29] is a peer-to-peer distributed hash table (DHT) where the nodes are represented as leaves in a binary tree. The nodes in the network have a node ID in the keyspace, in which the key is a 160-bit quantity. The keys have also the opaque property, which means that the key is represented in a way in which there is no direct access to the key material that forms the key.

The nodes are positioned in the tree in an structured manner. The location of a node is given by the shortest unique prefix of its ID. Therefore, when a node wants to be added to the network, according to its unique prefix, the binary tree is traversed and a leaf node is created for the new incoming node. It can be seen in Figure 2.5 how the node with prefix 0011 is located in a leaf node (represented as a black dot) according to the unique prefix.

Moreover, in order to decide the contacts of a node, the tree is successively divided. From the perspective of a given node, the binary tree is divided into multiple lower subtrees that do not contain the node. In Figure 2.5 these subtrees for node with prefix 0011 are denoted by gray ovals. It can be seen that the selection of the subtrees is done by, starting from the whole tree, considering the half of the tree not containing the node. This is successively applied in the next subtree, in which the half that do not contain the node is considered as another subtree, and so on and so forth until getting all k buckets (or k delegate nodes) in each subtree.

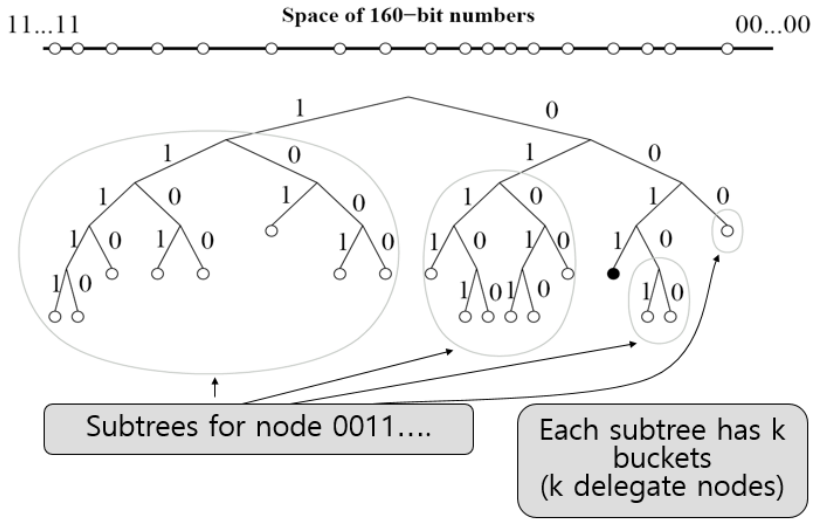


Fig. 2.5 Kademlia binary tree example [29].

Given this set of subtrees not containing the node, Kademlia has the important property to guarantee that that any node has at least one contact in each of its subtrees (as long as that subtree contains a node). With this contacts, a given node can locate any other node by its ID. The process of finding the node is illustrated in Figure 2.6, where node 0011 wants to find node with ID 1110. Node 0011 successively contacts the *best* node it knows, finding contacts in lower and lower subtrees until the target node is found.

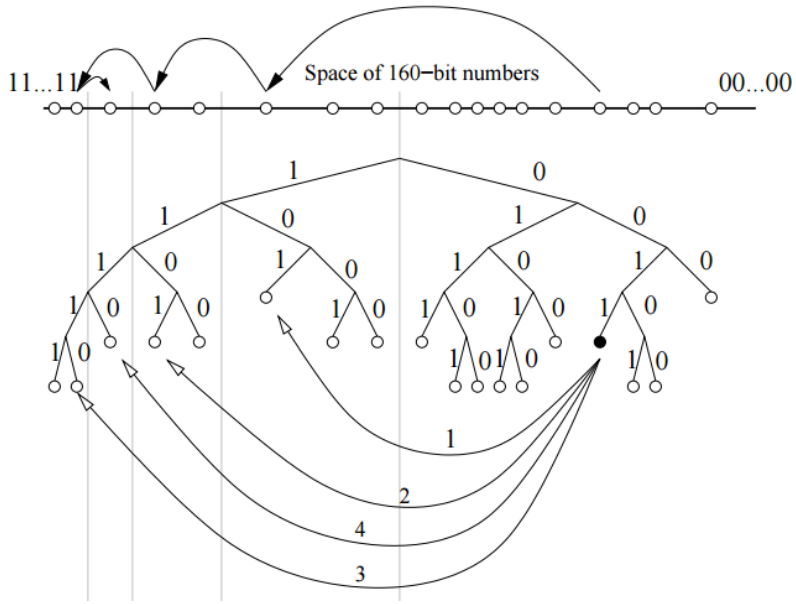


Fig. 2.6 Kademlia locating a node by its ID [29].

When in the last example, node 0011 contacted the *best* nodes, we are referring to the closest nodes given that we want to reach node 1110. Therefore, we need a measure of distance between nodes. In fact, one of the most important features of Kademlia is how this distance is measured.

In order to get the value of such metric, two node ID's or a node ID and a key are XORed[3] and the result is the distance between them. With the symmetric property of XOR, in Kademlia, the lookup queries that participants receive are precisely from the same distribution of nodes contained in their routing tables. Therefore, this property enables Kademlia to learn useful routing information extracted from received queries. If this property was not present, Kademlia would not be able to do that, just like Chord [42] cannot. Thus, this leads Kademlia to behave in a more intelligent way, selecting rules based on latency and even sending parallel, asynchronous queries to nodes that are equally appropriate.

[3]For each bit, the XOR function returns zero if the two bits are equal and one if the two bits are different.

It is important to note how the XOR metric captures the notion of distance in Kademlia's binary tree structure[4]. Namely, in a tree that is completely occupied, the distance calculated by the XOR metric corresponds to the height of the smallest subtree containing both the node IDs we are measuring. On the other hand, when the tree is not fully populated, the nearest node to a certain ID is the lead node whose ID shares the longest common prefix of it.

Taking the basic approach of many DHTs and using an XOR metric for distance between points in the key space, Kademlia simultaneously achieves many benefits that other DHT do not such as:

- The key lookups have the interesting and beneficial side effect of giving configuration information. This leads to a minimisation of the number messages that a node has to send to learn about another node.
- Proximity routing is achieved as with the knowledge that nodes have due to the symmetry, they can choose nodes with low latencies.
- Fault tolerance and concurrent execution is possible by means of parallel and asynchronous queries to the list of nodes most recently seen.
- Handles denial of service attack by using the algorithm that records the existence of the nodes.

The XOR metric that allows a symmetric treatment of the tree along with those provided benefits, specially the fact of being able to execute parallel queries and its resilient design, makes Kademlia a perfect element for our proposal implementation.

[4]Note that although XOR is a valid metric, it is not Euclidean.

Chapter 3

Related Work

Edge-centric computing has been recently proposed as the natural evolution of peer-to-peer (P2P) systems. As described in chapter 1, one of the major claims of this new paradigm is to move the control to the edges of the system (human-controlled) in order to preserve user's privacy. Edge-centric systems may combine centralized and decentralized components in order to overcome the limitations of P2P systems.

In this new paradigm, TallyNetworks is one of the first Internet distributed services following the edge-centric paradigm. TallyNetworks combines centralized (Authenticator) and decentralized (TallyBox one-hop overlay) components to fight against security treats (like the Sybil attack) and to simplify architecture services (opinion voting and opinion retrieval, user membership, global knowledge of the routing table).

However, there are a lot of previous work that is noteworthy to point out as they have served us as inspiration, model or proof of correctness. In the following sections, we will describe the main topics that helped or inspired us in someway beyond the previous described background.

3.1 E-voting systems

Although TallyNetworks differs in many aspects with E-voting systems as it will be seen during this introduction to the traditional and common definition of an E-voting protocol, we cannot ignore the huge influence that such systems have had on our proposal.

If you think about your local elections, you will see that the way E-voting works is very similar to how in-person elections works but from an electronic point of view. In fact, E-voting stands for electronic voting, which is defined as the action of voting using an electronic procedure [48]. Such procedure can be applied to assist both the task of casting

and counting the votes. In the following lines, we will describe the main ideas and steps behind electronic voting.

Usually three main protocol steps are defined in an e-voting system, besides to an extra step when the protocol is finished:

1. **System Access Control Process**: In this step the voter is authenticated in the system so that it can be authorised to vote. In order to accomplish this step, two procedure are encompassed:

   - *Identification phase* : The user wanting to vote will claim an identity given by some unique identifier for a user. This is also known as the registration phase.
   - *Authentication Phase* : In this phase the voter provides certain credentials to prove her identity. Usually, and it will also be the case in our proposed system, this is done by a public key.

2. **Voting Process**: In this step, the actual vote of the voter happens. Such vote must guarantee the privacy of the user, so using public key encryption or any other system that permits to the user cast the vote without reveling her option, the vote information is sent as an encrypted text to the election server. In this phase, usually (and also in the system we propose) some replication is done to provide robustness to the system.

3. **Collecting or Tallying Process**: In this phase the counting of the votes is performed in such a way that the system unlinks each vote from the person that casted it, so although being able of opening the votes or aggregating them and obtaining the final result, the system cannot learn from the option of the user. In most of the voting systems, there is a scheduled time of voting that when is finished is when the counting takes place to give the results. However, we will later see that our method allows the voting process to be always open, while votes are being count. Also, in most traditional e-voting systems, voters can only consult the result at the end of the process when the tallying is finished. In our case, we propose a model in which any participant can retrieve the results of the voting at that moment. This will make our architecture not suitable for some applications such as political elections but very useful in other field such as the reputation of users in the e-commerce. Another aspect that has to be taken into account is that when the tallying is made, the system has to validate that the signatures of the votes are from a valid participant.

4. **Validation Process**: Once the protocol finishes, all participants or an external entity must be able to ensure that the process counted all the votes and that the final result is correct.

This typical steps in an e-voting system has also some common desired properties which are the following described in [2] among many others, as such properties are common to any e-voting system:

- **Authentication**: only people in the electoral roll can vote.
- **Unicity**: every participant can vote once at most.
- **Privacy**: votes can not be related to voter identities.
- **Fairness**: no partial results can be revealed before the end of the voting period.
- **Verifiability**: correctness of the process can be checked.
- **Uncoercibility**: nobody can prove that a voter voted in a particular way.

The described protocol and the way such properties are fulfilled is a wide field that has been considered in different scenarios. But basically we can distinguish two different approaches: centralized voting and remote voting.

In the following sections we will introduce some different solutions based on this two approaches.

3.1.1 Centralized E-voting

One of the most common and studied environments is a centralized approach. In this approach, we can distinguish three different schemes or paradigms as a solution to the remote voting challenge:

- **Mix-type:** In this paradigm a voter signs her ballot and then casts after encrypting it. Once the time for voting ends, the central entity that received all the ballots, shuffles and re-encrypts them. This process is also known as mixing and the aim is unlinking the votes from the user that casted it.

  Some examples of this approach are [39], [22] or a more recent one [28]. Although it is a clever an useful solution, it would clash with some problems when moving from a centralized environment to distributed one. Votes should be mixed among different nodes and as they are not trusted at all, we should also replicate the votes in order to make sure all of them are counted while we ensure that they are not counted more than once. Furthermore, managing all the keys and reencryptions required would make things even more complicate. These are the main reasons why we decided to explore other options.

- **Homomorphic tallying:** We will first introduce the concept of Homomorphism. In cryptography, a system is homomorphic if allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

  In homomorphic tallying schemes, participants cast their ballots encrypted under some public key cryptosystem having a homomorphic property as it would be the case of ElGamal encryption which allows additive homomorphism.

  Votes are casted and sent to the trusted entity that will aggregate all the encrypted votes in order to obtain a single ciphertext with the result of the election. Once the voting process period ends, the ciphertext can be decrypted in order to obtain the final aggregated result.

  Notice that in this system, votes must be encoded in such a way that the final tally can be recovered from the cleartext of the aggregated ballots, for example, 0 means no, 1 means yes or also we could use a vector of options.

  As votes are aggregated in ciphertext, we must ensure that its content is valid without being able to know the real content. This is why we usually need to use zero-knowledge proves to make sure the ballot has been composed properly. Zero-knowledge proves [49] allows that one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true.

  Some examples in the literature using this paradigm are: [7], [21], [34]. As in the previous case, those are very clever solutions that achieve solving the problem of remote voting offering all the required guarantees, however, moving this model to a distributed environment it is not an easy task. If votes must be aggregated in a distributed fashion, we must ensure with some mechanism that each vote is aggregated only once. Furthermore, key management will add complexity as we will require to provide reencryption schemes in order to distribute and aggregate all votes in different nodes. In addition to this, as we cannot trust all nodes of network, we should do zero-knowledge proofs to each vote that we receive from any other node (single or aggregated votes), not only when it is casted by the user, which would add an extra computation overhead besides reencryption. For all these reasons, we decided to explore other possibilities.

- **Blind signatures:** The concept of blind signature was introduced in the Background section of this work. The idea behind an e-voting system that relies on blind signatures is that a voter authenticates against a trusted authority which is responsible for checking

that the voter can participate in the election and she has not voted before. In that case, voter's ballot (the encrypted vote) is blindly signed by that authority. The polling station only accepts ballots that have been properly signed by the authority. When the voting period is concluded, ballots are decrypted and tallied.

Some examples of this approach are [5] and [16]. However, although being very good solutions for a centralized approach, again, distributing such central entity would require constructing a mechanism to synchronize all nodes in order to be able to verify if the voter has already casted the vote or not independently of the node he is trying to cast the vote to. Furthermore, votes must be encrypted and blind signed, so it is clear we will have a lot of work with key management as the authority is distributed and each of them will have a different key. Furthermore, as all nodes in the network will be blind signing the vote, but also counting the votes, we will need a way to anonymize the user when casting the vote without losing the capability of guaranteeing one vote per user.

The inception of our proposal inspired by blind signatures

The fact of realizing that any node should be able to decrypt the vote and blind signing it in a distributed environment using blind signed ballots inspired us to become with our proposal. In a blind signature scheme, users encrypt their vote in order to avoid the authority or the polling system learning about our vote so they can link the vote with the user. Nevertheless, if all the nodes will be able to decrypt the vote, it doesn't really make sense to encrypt it, as we are not really doing nothing in a practical sense.

In fact, we encrypt the vote to achieve privacy, but if encryption is not providing this privacy because of the distributed approach, we must achieve privacy from another point of view. A feasible option is achieving privacy by anonymizing our identity when casting our vote. If we are anonymous, we won't need to encrypt the vote.

However, we should sign the vote in order to provide integrity. In consequence, the fact of signing the vote means that we need an identity to sign it, but also we need such identity in order to provide one vote per user guarantees. The way of obtaining an identity without revealing the real one is using a pseudonym as stated during the background section.

By using a pseudonym with which we will sign the vote, we can achieve privacy without compromising integrity if we can guarantee that such pseudonym won't be linked with our identity. This can be achieved by using the same idea behind the fact of

not learning about the vote, blind signatures. we can generate a pseudonym and obtain a blind signature for it. A blind signature will only be issued once to the same person in order to guarantee one vote per user, but as they don't learn about the pseudonym, our privacy is not compromised.

In our proposal, which later will be described in detail, an authenticator component uses blind signatures to create an anonymous pseudonym. In this case, such pseudonym will be a public key and it will be used for voting and participating in the network. By combining blind signatures and anonymous channels, we provide a level of pseudoanonimity that is by far sufficient for the problem of distributed management of personal opinions.

We want to make a final note about comparing TallyNetworks to traditional e-voting systems as used in public elections. Such systems are normally based on centralized trusted components [33] managed by a public institution. The security requirements in these systems are extremely high in order to have an acceptable e-voting system. For example, voting from mobile devices cannot be considered without secure hardware devices. This is one more reason why TallyNetworks cannot be compared directly to them besides the previously mentioned reasons such as the need of offering real time results or voting periods with no end.

Besides the centralized approaches that we have presented, there is also work around distributed remote voting systems and we will introduce in the following section the main ideas behind them.

3.1.2 Distributed E-voting

Yet a more interesting topic we will study for this work is when the remote voting is organized in a distributed architecture. This provides some interesting benefits such as more robustness against denial of service attacks, not relying on a trusted entity that controls all the election and empowers election participants, which is close to the goals we aim to achieve.

Distributed E-voting is a trending topic nowadays with the disruption of cryptocurrencies, nevertheless, there are not known real initiatives implementing or using distributed e-voting. Besides to cryptocurrencies, other proposals can be found in the literature using other cryptosystems as we will briefly describe.

Chan et al. [3] proposes a system in which each member publishes its own encrypted vote in any server in such a way, once the votes are aggregated, the collaboration of all participants is required. Then, once all of them have voted, each voter will retrieve all the votes and will tally them using the homomorphic property provided by ElGamal encryption. They will also

share their random exponentiation in order to be able to validate the result of the election all together. However, they don't provide a method for registering voters because they assume a small group. Furthermore, the fact of collaboration between users to obtain the final result can be an scalability threat.

Another approach based on broadcasting and using homomorphism is the one proposed in [36], however, this approach has been thought to use it in very little communities as all the participants contributes to the key generation and decryption process of the votes, so they all have to be online when key generation or decryption happens in order to obtain a correct result.

Beyond these proposals we can find the technology that seems that will have an important roll in the world in the following years because of the tremendous interest that has awaken, which is blockchain. Blockchain can be used in many fields and distributed voting is one of them. In fact, there are some projects such as Follow My Vote [1] that tried to build a solution for e-voting based on blockchain, however, it seems that is not being actively implemented as their public repositories doesn't have activity since the end of 2016 at the moment of writting this work.

Although blockchain is being very trending, it doesn't seem that exist any public implementation of e-voting, but a lot of people is speaking about it as you can find press analysis, blockchain experts and even some goverments as the Estonian [2] pushing blockchain e-voting and speaking about it.

Despite of this trend around blockchain, with our proposal, we wanted to explore alternatives to a blockchain voting protocol. However, we will briefly introduce how blockchain works and how e-voting would fit in it, because as it will be seen, we can find some coincidences between our proposal and e-voting based on blockchain.

We can define blockchain as a distributed databases that registers transactions in such a way they cannot be altered because they are tied forming a chain that is distributively formed and maintained by all the users of the network.

Everything begins when a user wants to register a new transaction. She will compose the transaction and will use his secret key to sign it. The public key associated to this secret key will be her identifier in the network or her pseudonym. The transaction is broadcasted to the network and all nodes in the network will validate such transaction. Once it is verified, it is combined with other transactions to create a new block of data. Then, the block is added to the existing blockchain in a way that is permanent and unalterable.

[1] For further information visit: https://followmyvote.com

[2] Estonian goverment is said to successfully complete a pilot program with e-voting based on blockchain: https://www.cyberscoop.com/nasdaq-estonia-evoting-pilot/

In this sense, the transaction will be the vote and the validation process will consist on verifying that the vote is valid. The block chain will contain all the casted votes. Privacy is achieved thanks to the pseudonym. The main problem with this is that anybody can generate a new pair of private and public key that are used as pseudonym. The private key would have to be verifiable tied to one individual's identity, and that individual's identity would have to be exclusive to all other entities in the voting network. Although the voting system could implement a 1 vote per private key when validating the vote, the pseudo-anonymous nature of blockchain network makes nearly impossible to know if an individual has generated two private keys. For this reason, we would require a way to ensure that only verified pseudonyms can participate in the election, probably with a trusted entity.

As it can be deduced by this brief explanation, blockchain is relying on pseudonyms, the same as our proposal as we introduced in the explanation of the inception of our idea. Furthermore, an access control is required, which was also a problem in our proposal until we added an authenticator component to do this task in order to decide which members are accepted in the network.

However, implementing blockchain on mobile devices it is not feasible because of the huge computation required and also because of the huge amount of required bandwidth as it would drain the battery and end with the available data for the user. For this reason, we should rely on intermediate servers, similar to our TallyBoxes.

The main difference between blockchain and our proposal is that blockchain ensures that everybody will receive all transactions thanks to the permanent and unalterable blockchain that nodes will be able to retrieve and construct. However, it cannot ensure that the broadcast to start the validation process of a transaction is reliable enough to reach the required majority as it uses a gossiping[3] algorithm. Therefore, the transactions that are present in the blockchain are the ones that where properly broadcasted. In contrast, our proposal aims to ensure that the 100% are reached when broadcasting the vote, so they all will be able to check if the vote is valid or not and count it without other validations.

We could say that our proposal is a simpler approach than blockchain as it doesn't require as much communication as blockchain and the required computation and the size of transactions is smaller. However, as it is simpler there is an important difference in terms of security. Blockchain is secure by design because of its transaction verification and block chain and if the broadcast is not 100% reliable, the nodes that couldn't be reached, will acknowledge the vote once it is published in the blockchain, while our solution is statistically secure in the sense that it relies on ensuring that the message will reach the 100% during the

[3]The main idea of a gossiping algorithm is that there are a bunch of nodes, and when one of them acknowledges something, it tells it to a few nodes that are near (connected with), and then those nodes tell it to a few more nodes and so on. Eventually everyone knows about the original information (transaction).

broadcast phase and if any node cannot be reached, it will be later synced with the rest of nodes thanks to a synchronization phase.

However, taking into account the purpose of our system, we provide a level of pseudoanonimity and security that is by far sufficient for our purpose.

3.2 Distributed reputation

The system we are going to propose in the next chapter, TallyNetworks, is applicable to distributed reputation. Reputation systems are used in different electronic services and platforms such as e-commerce, in which the users gain a trust from the other users. This makes the platforms more reliable for the users as they can check the reputation of other users, for instance sellers in the e-commerce field.

Due to the large data that such platforms usually have to deal with, distributing the reputation calculation is an essential aspect. This, however, introduces a further challenge, since the distribution of the reputation measuring will trigger some security threats that should be carefully studied. In fact, our proposal in the next chapter aims to provide a distributed application of reputation in the platforms mentioned above in a secure and scalable way.

Previous work has been also done about this field that is worth studying.

An interesting study [24] endorses our claim of the importance in reputation mentioned before. They pointed out how trust is an essential component not only in real-life social interactions and business relationships but also in the on-line environment. This fact together with the explosive growth of electronic communities, make the building of trust of the entities in those communities an essential factor for its functioning. For instance, in the field of electronic commerce, the building of trust into on-line vendors and their items is vital for its success. For this reason there is the need for a reputation management. The reputation of an entity is defined as the average trust of all the other entities towards that entity. It can be noted how the reputation has a global connotation in contrast of the trust. Therefore, following the previous example, a seller with bad reputation is prone to have little purchases.

Different algorithms for reputation management in a distributed environment have been studied (see [20] for a survey). For instance, in EigenTrust [23], all the nodes in the network cooperate to compute and store the reputation of all the nodes so that the computation, storage and messages that have to be carried for a node is minimal.

However, completely distributed or peer to peer approaches in a mobile first world like the one we are living may clash with some limitations. Participating in a peer to peer network with the above mentioned characteristics using an smartphone may drain the battery, especially in very active networks, because of the constant required connections to establish

and calculations to perform. Also, it would probably consume more data than the desired by the user because of the number of messages that must be sent or received.

On the other hand, if users decided to be selfish and only connect to the network when they want to vote or know someone's reputation in order to avoid the previous stated consequences, a high churn would be produced and the stability and reliability of the network would be compromised.

In our proposal of TallyNetworks we want precisely to overcome such problem by means of using the TallyBoxes, enabling the user to use a more or less stable device (server) to perform the operations for her without the need of being connected all the time but instead retrieving information from the server when needed. Other similar protocols such as PeerTrust [50] encounter the same problem as their focus is on peer-to-peer networks.

3.3 One-hop structured overlays

Another relevant related work is the literature on one-hop structured overlays [19], as we will used them in our proposal to easily access to the different participants without the need of doing multiple queries to reach our destination as some systems, such as Chord, do.

Such overlays are highly suitable in networks of relatively stable peers like the ones proposed by edge-centric computing. In this case, the widespread adoption of stable home appliances (storage and compute sticks, media centers, nanodatacenters) and the improvements in residential bandwidth can create very efficient Internet services like the one proposed in this thesis.

Therefore, by using one-hop overlays in the membership of TallyNetworks, nodes will be able to retrieve all membership information in one step. This leads to a reduce of communication and also to a faster results. Moreover, it is more scalable and it can aggregate information in a trustworthy way.

An important difference with previous related works on one-hop overlays is the algorithm employed for the efficient management of a huge up-to-date routing table. Previous works [19] aim to reduce the bandwidth imposed to nodes due to the event dissemination of active peers in the routing table. In our case, our global membership service will only send events about permanent joins or leaves from the network, and not about transient churn. Our novel approach efficiently combines a redundant broadcast algorithm for Kademlia with a syncing protocol.

We analytically study the relationship between redundancy in the broadcast and syncing phases to ensure that all active peers have a fresh global view of the network. Transient

churn, malicious and unresponsive peers can be ignored or bypassed easily while ensuring one-hop query services.

A simple alternative to our membership service could be a gossip protocol for member dissemination. In fact, our syncing phase is equivalent to a pull gossip protocol, but the combination of broadcast and syncing guarantees better convergence time in large steady-state networks. We advocate for an adaptive membership protocol that can rely only on syncing in fast-growing periods, and resort to broadcast+syncing when the network is more stable.

3.4 Networks of edge web servers

Finally, previous works like [40] have linked networks of edge web servers using structured overlays by creating an open, decentralized infrastructure to enable Web servers to use their spare capacity to filter out, aggregate and disseminate Web content in a scalable and timely manner. Interconnecting stable web servers using HTTP in structured overlays offer interesting value-added services to applications such as indexing or efficient content dissemination.

As stated in [40], our implementation also interconnects edge servers in a structured overlay. However, in this proposal, our efficient one-hop overlay considerably reduces the communication overhead for the servers thanks to the adaptive broadcast/sync membership model.

In particular, indexing (put, get), naming, aggregation or efficient content dissemination are interesting application level services that can be built on top of these infrastructures.

Chapter 4

TallyNetworks

Nowadays all our information is in the hands of very big companies. Some examples are Google, Facebook and Amazon, to name but a few. With all their gathered information, they trade with advertisers [12] the value of all such data. This makes us highly dependent on those companies in a centralised scenario in which all is delegated to them.

This leads us to propose another model focused on offering a system to let people express their opinions such us like/dislike or polls, but with a key feature, they must be able to express their opinion in a private/anonymous way, so we can acknowledge the opinion of the community, but not the opinion of each member of the community.

Furthermore, we believe that key personal and social communication services should be decentralized and human-driven. This is why we propose a decentralized solution to empower people. The concern about companies controlling your personal information has been previously stated such as in [5], which also proposes some different approaches to address the problem.

This master thesis describes a new architecture, TallyNetworks, that allows to preserve your opinions/vote's privacy in large communities based on the new introduced paradigm of Edge-centric computing.

This proposal contrasts with other solutions such as the ones introduced in previous sections based on centralized or distributed e-voting protocols, reputation systems or even voting systems based on blockchain. TallyNetworks doesn't present itself as a better solution than the ones introduced, but an alternative solution presented by us which addresses our concerns about the control of our opinions and our privacy.

In this section, we explain the key insights of our idea and the overall architecture of our solution, the life-cycle of the system and our novel global membership protocol based on Kademlia.

4.1 Objectives and main idea

The objective is to design a distributed architecture and a communication protocol with the two following main principles in terms of robustness:

- The distributed architecture has to be resilient to malicious nodes based on the Edge-centric computing paradigm.
- The communication protocol has to be resilient to malicious node attacks for such architecture.

Recall that our aim is to build a system that allows people to participate in any kind of poll or rating without disclosure of their real identities. However, not only we want to achieve such objective but also we want to ensure the integrity of the result of any poll in the presence of malicious behavior. Moreover, what is more important is that we want to provide *end-to-end verifiability*. This means that both the reception and tallying of votes is correct and its correctness can be verified from the cast of the vote to the final result. As in most of poll or voting systems, the user can only vote once and the system must be able to guarantee this.

In fact, the desired properties of our system are very similar to the ones described in the E-voting explanation, except for fairness, as in this case, we want to be able to offer dynamic and real time results as like/dislike systems or opinion systems scenarios always work in a real time or dynamic way.

Instead of a pure cryptographic solution, we propose *TallyNetworks*, a system that meets this challenge through a novel integration of cryptographic techniques with a one-hop Distributed Hash Table (DHT). Thus, we can benefit of the strength of both cryptography that provides properties such as privacy and security and distributed systems that provides properties like scalability or robustness.

For a given poll, the basic idea is to leverage the underlying one-hop DHT to assign the task of tallying to a subset of TallyBoxes with enough redundancy to ensure the correctness of the result. Moreover, in order to guarantee the correct delivery of the votes to the responsible TallyBoxes, we will combine a broadcast algorithm with a pull based approach, recasting our problem as a secure distribution of votes.

4.2 Security properties

Before starting the description of TallyNetworks we recall the security properties that our system needs to have. In fact, this is an important research area in its own right [41, 43]. The most important security properties can be grouped as follows:

- **Integrity**: This is concerned with the avoiding of alteration or corruption of the opinions submitted. Additionally, intruders should not interfere with the process. More specifically, integrity implies that a submission is cast as intended, recorded as cast and counted as recorded. Integrity entails honest behavior, in our case, an honest behaviour of the minimum users of the system depending on the setup. It also entails collusion resistance.

- **Privacy**: This property is aimed at ensuring that votes are cast anonymously or in our case, pseudo-anonymously. In other words, it means that it is not possible to associate an opinion with the corresponding real user identity (untraceability).

- **Verifiability**: This criterion is related to integrity and it refers to the openness of the system to formal and practical inspection. This is divided into two verifications:

  - *Individual verifiability*: It should be possible for users to check that their opinions were correctly recorded.
  - *Universal verifiability*: It should be possible for users to check that all the opinions were processed and counted correctly.

  It is believed that with enhanced verifiability, users have more confidence in the system.

- **Robustness**: This last property is defined as the resilience of the system when it is subject of external or internal malicious attacks. It should be also resilient when cheating behavior is detected or even when system malfunctions occur (partial or non-partial). The system should operate as expected in abnormal conditions or in a hostile environment. Some examples of being robust are: resisting a denial of service attack thanks to the distributed nature of the system, ensuring that all honest nodes receives a casted vote even if a malicious node drops messages or if a node in the network is down...

In Chapter 5 we analyse each one of the previous properties for our novel system TallyNetworks that is explained in the following sections.

4.3 Entities of TallyNetworks

In this Section our actual presentation of TallyNetworks starts with the description of its entities. Our proposal of TallyNetworks is composed by three entities involved in the system that are described below:

- **Participant:** A registered user who can emit opinions (`like`/`dislike`) or participate in public polls anonymously from a mobile terminal. It can also query the system about the current state (votes, opinions) of specific polls or items.
- **TallyBox edge server**: It is a node of the TallyNetwork overlay. It can cast votes but also retrieve and query them using the one-hop DHT. It receives votes from participants and redirects them to the appropriate TallyBoxes using the opinion identifier. The responsible nodes will then count and store the votes if they are valid according to their credentials.
- **Authenticator server:** It authenticates participants and TallyBoxes based on an admission policy and blindly signs their credentials. For participants, it can check whether they are members of the community or real authenticated users. When a TallyBox edge server is accepted, then the Authenticator server will assign a unique identifier to it, in order to avoid Sybil attacks [15].

4.4 Security Threats

Our system has to provide security features according to the potential malicious participants. For this reason, it is useful to identify which are the security conflict situations that may appear in the system and how our system should overcome them. The two main scenarios with its corresponding expected system behavior are the following:

1. We assume that the goal of a malicious participant is to try to tamper with the poll results, for instance, by emitting multiple votes to favor some option, or even by emitting contradictory votes for the same poll.
    - *Expected behavior*: Our system will have to handle these situations in order to ensure one vote per user and that all participants have seen the same vote counting result.
2. Further, we assume that a malicious TallyBox can drop messages, flood the network with fake messages or try to disconnect other TallyBox servers from the network.

- *Expected behavior*: Our system must mainly prevent the loss of votes in addition to thwart Denial-of-Service[1] (DoS) attacks and overlay partitions.

4.5 Protocol steps

In this section we define the steps of the protocol, that are different for participants than for TallyBoxes.

In Figure 4.1, the protocol steps that a TallyBox follows are depicted and they are explained below:

- **Obtaining the node identifier:** In order to register a new TallyBox in the network, the first step is to contact the Authenticator server and request a node credential. The Authenticator can accept or reject the request taking into account the admission policy. If it is accepted, it will receive a signed credential that includes the node identifier and the URL by it will be accessible to the rest of nodes.

- **Entering the network:** Due to the fact that the network of TallyBoxes is a one-hop overlay, a joining node only needs to contact a group of TallyBoxes in the system to construct the entire one-hop routing table and obtaining the information of the active polls it is responsible for. Notice that it contacts a group of nodes in order to establish a consensus from the different one-hop routing tables and information it receives from the contacted nodes.

  Thereafter, the new node will broadcast its join request signed with its credential to the rest of them, so all the nodes in the network will know about the new join to add it to their respective routing tables.

  It is noteworthy that all this procedure is transparently handled by the underlying DHT itself.

- **Participating in the network:** It basically consists on forwarding messages or requesting updates during broadcast and syncing phases. Every TallyBox can fine tune its activity in the two phases of the protocol depending upon its resources. For example, a weak node might decide to avoid participating in the broadcast algorithm, and only periodically synchronize on a per-day or per-week basis.

[1] A denial-of-service attack (DoS attack) is a cyber-attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled [30].

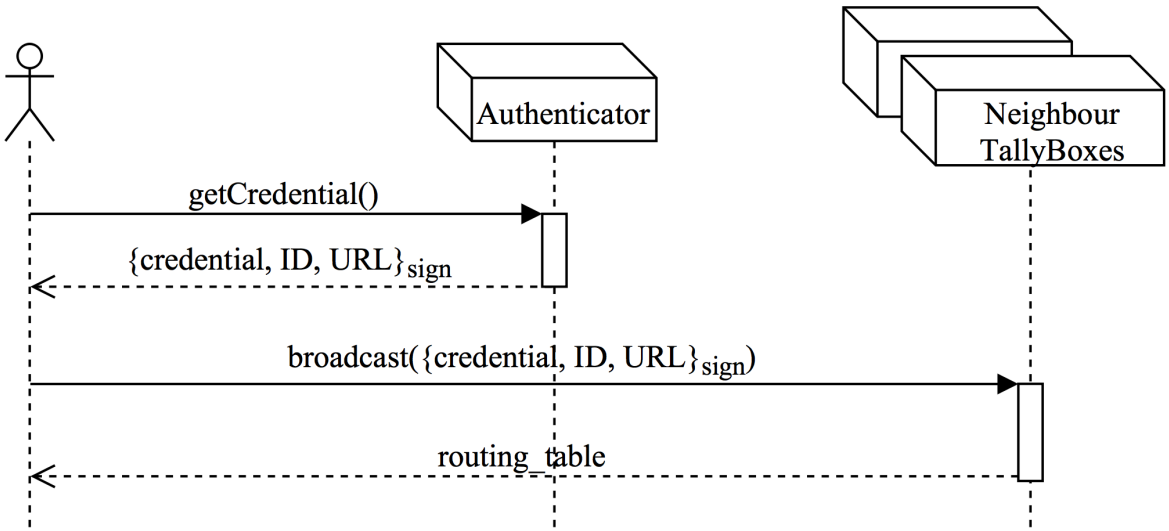


Fig. 4.1 TallyBox Protocol

On the other hand, a participant follows another protocol steps that are shown in Figure 4.2 and described below:

- **Obtaining a user credential:** In order for a participant to register in the system, the first step is to contact the Authenticator server to request a user credential. To achieve that, the following steps are performed:

  1. The joining participant sends his public key PK in blinded form $\mathscr{B}(PK)$.
  2. It can be admitted or rejected by the Authenticator depending on the admission policy.
  3. If admitted, he will receive a blindly signed credential $\{\mathscr{B}(PK)\}_{sign}$ from the Authenticator to participate in TallyNetwork.
  4. Upon reception, the joining participant will unblind the signature to get $\{PK\}_{sign}$, which will be, from now on, his public credential or pseudonym to participate in the network. Notice that thanks to the blind signature, the credential is signed by the authenticator, but he doesn't learn about the credential, so it cannot be linked to the user.

- **Voting process:** The participant can now use an anonymous channel, for instance Tor [14] which has been previously described in section 2.2, to cast a vote in any TallyBox using his public key PK as pseudonym. For robustness reasons, the vote is cast to more than one TallyBox. In this way, if a TallyBox is unavailable or decides to drop

messages, other TallyBoxes will ensure that the vote is properly distributed to the network.

To actually cast a vote v, the vote itself, its signature $\{v\}_{SK,sign}$ with the participant's private key SK, and the authentication credential $(PK, \{PK\}_{sign})$ must be sent via the one-hop overlay to the TallyBoxes responsible for the poll. For example, if a participants wants to add a `Like` to "Hans", this `Like` vote will be addressed to the TallyBoxes responsible for the key "Hans". The number of nodes responsible for each key can be configured to ensure the robustness of the voting process.

Notice that the vote is signed in order to guarantee its integrity. On the other hand, the credential is sent in order to link the vote to a credential, so we can guarantee one vote per user/credential, but also in order to check the correctness of the signature with the credential. Finally, in order to guarantee that the credential and the vote corresponds to a member of the network, we use the signature of the credential from the authenticator, which guarantees the belonging to the network.

- **Tallying:** Unlike traditional voting systems, the voting period can be always open because of the nature of public opinions in social environments. Therefore, a participant can retrieve the current state at any time.

  Tallyboxes will receive votes for the polls they manage. For each vote, they will verify the integrity of the vote and that the signature comes from a valid participant. Then, they will check that the vote is not a duplicate one, which means that they haven't already accounted the vote in their list. In order to keep the vote counting up to date, the TallyBoxes responsible for the same key will also be "in sync" so that vote counting will eventually converge.

  Notice that in case a malicious participant cast two votes for the same poll with same values, the second will be ignored in order to guarantee the property of one vote/one value. However, if he cast two votes but both have different values, then both will be discarded in order to ensure that all the nodes will respond with the same result.

  It is also noteworthy that the fact of having a list with all the votes in each responsible node, allows the participant to retrieve the entire list and check if her vote appears in the list or if the public result has been properly calculated.

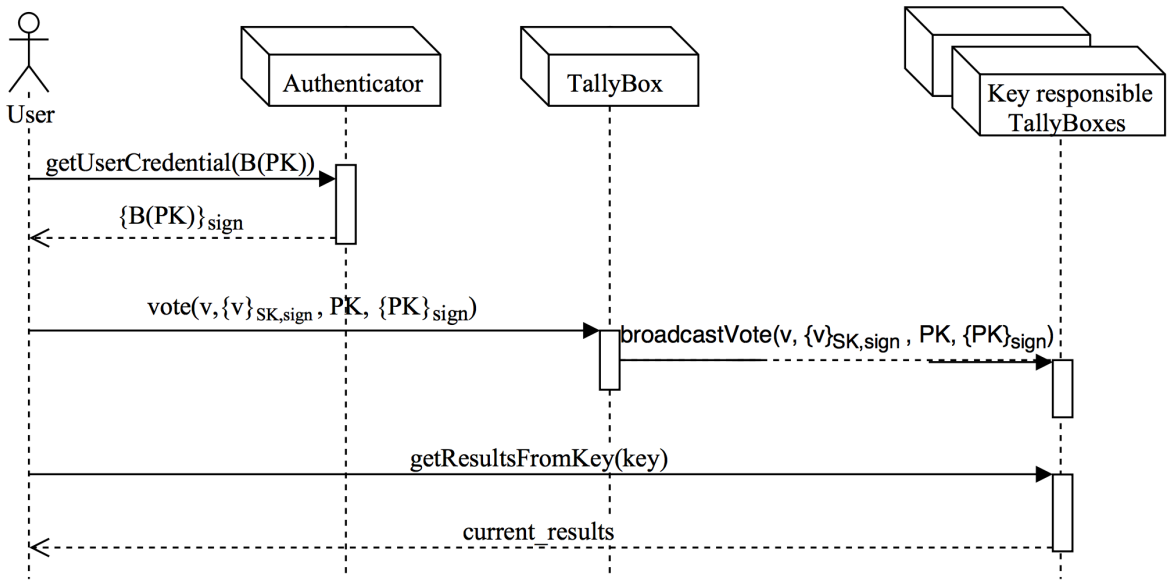


Fig. 4.2 Participant Protocol

4.6 One-Hop Architecture: Membership

The objective of this thesis is to build a system capable of satisfying all the goals listed in the introduction section and argued throughout the Background and Related Work Sections. In order to do that we proposed to implement an efficient DHT of stable TallyBox edge servers because of its scalability and reliable storage of distributed information properties.

On the grounds that servers do not maintain active connections (HTTP), they will have enough resources to be able to maintain in disk a huge amount of contacts in the network that would not be possible otherwise. This is clearly in line with previous one-hop or two-hop overlays that maintain big routing tables that has been previously mentioned.

Having a one-hop overlay, allows the system efficiently contacting any node in the network with only one hop. Therefore any possible algorithm can be implemented over this overlay in an optimal way such as our broadcast and sync phases which contacts the minimum number of nodes in order to guarantee all the desired properties.

The fundamental difference of our approach is that we only handle permanent joins or leaves to the network due to the stable nature of TallyBoxes, and not transient churn. This avoids active checks or keep-alives to detect and propagate the availability of nodes in the system. As it is explained below, our network is designed with sufficient redundancy to overcome not only transient churn but also malicious participants.

Kademlia, which has been previously introduced in the background section, has been chosen as the structured overlay to use for our implementation. The principal reasons are

its resilient design and the ability to issue parallel queries. In order to maintain the routing tables up to date, we propose a novel Kademlia CAST & SYNC algorithm (see Section 4.7 for full description) that leverages an existing broadcast algorithm over the Kademlia tree overlay.

Notices that our algorithm is adaptive to the size of the network and capabilities of each TallyBox. Additionally, it permits the configuration of the redundancy level across the broadcast and synchronization phases to fit with the TallyBox resources, but also to ensure the desired level of robustness of the network.

4.7 Kademlia CAST & SYNC

In this section we describe our novel membership protocol that aims to be adaptive to the size and also persistent churn of the network. In fact, in a very small network, the broadcast (CAST) algorithm may become a star topology. On the other hand, in fast-growing networks, the broadcast traffic might be very costly. In that case, the synchronization (SYNC) phase should prevail.

The combination of CAST & SYNC algorithms is ideal for large steady-state networks. In this scenario, the broadcast algorithm will accelerate convergence time and interactivity, and the SYNC phase will guarantee 100% coverage.

The next two sections define the reliability of the broadcast and the syncing phase separately. However, we need to combine both to obtain the total system reliability.

4.7.1 Broadcast

The first phase of our proposal tries to arrive to the maximum of nodes in the minimum possible time. This can be achieved with a broadcast which in our case is built upon a previous Kademlia broadcast algorithm proposed by [9, 35].

This algorithm divides the key space using the Kademlia k-bucket routing tables. The main idea behind it is that the initiator of the broadcast will send a message to a contact or representative in each bucket. Then, recipients will forward it to their contacts or representatives in their buckets, but only to those within their own region. An example is depicted in figure 4.3 in which node number six starts a broadcast contacting a node in each of its buckets: 0,4,10 and 24. Then, each representative contacts a representative of each of its buckets, but notice that only contacts to those within their own region. For example, the region of 0 is 0, the region of 4 is from 4 to 5, the region of 10 is from 8 to 15 and finally 24

has a space defined from 16 to 31. Notice that the space is defined depending on the XOR metric.

This solution is the most natural and reliable way of dividing the key space in Kademlia for broadcasting. Moreover, it has been widely proven by the referred authors.

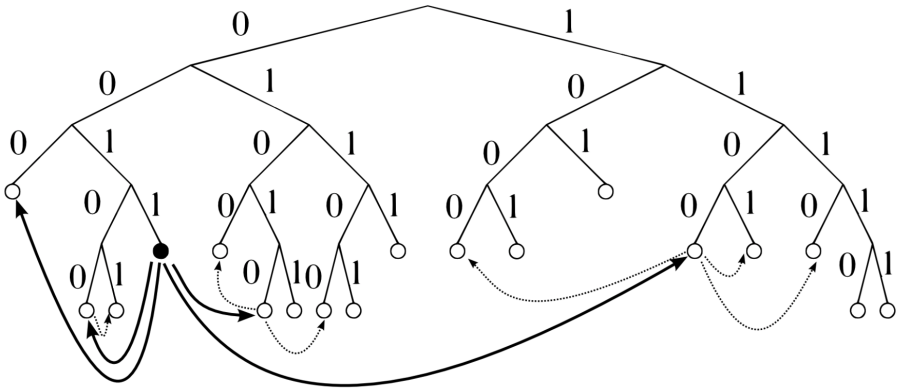


Fig. 4.3 Kademlia broadcast from node 6 [8]

However, a simple Kademlia broadcast cannot assure the total coverage of the network, because messages can be lost due to churn and malicious participants. In our case, because malicious and offline TallyBoxes have the same effect on the network, we will treat them equally.

Additionally, our system wants to overcome message losses. One typical solution to this problem is to increase the redundancy of the broadcast algorithm, so the message can arrive through different paths to the same destination. In order to achieve that, we will simply choose more than one representative contact in each bucket. This leads to a huge improvement of the reliability of the broadcast. Besides, the shortcoming is that its communication cost quickly increases. Therefore, as redundancy increases, the traffic grows with it, compromising scalability. However, a network that is handling popular opinions is very susceptible to be attacked, so high packet loss ratios should be tolerated, which means that we will need more redundancy. For this reason, we have to optimize the level redundacy we will implement in order to reduce the loss effects without compromising scalability as we will later discuss in the redundancy analysis section.

Below we describe, with specific numbers associated to messages broadcast, an example to depict the problem so that it can be clearly presented.

If each node had 160 buckets, the initiator of the broadcast would send $n * 160$ messages, where n is the redundancy level. In our case, the messages size would be: TallyBox server ID (160 bits) + IP address (32 bits) + port number (16 bits) + signature of the server ID (2048

bits), the latter depending on the chosen cryptographic system. With $n = 3$, only the initiator of the broadcast would send around 1MB. If this number is ported to huge networks, it is obvious that scalability will be severely undermined.

It is clear we must disseminate information efficiently and with high resiliency to packet losses, but also minimizing the communication traffic in the network. In order to reduce redundancy needs, we will complement the CAST protocol with a SYNC phase (explained in Section 4.7.2).

Redundancy analysis

Let us first study the analytical model presented and validated by [9]. Later, we will extend it with the SYNC protocol in Section 4.7.2.

To start with, let B be the number of nodes receiving the message over all nodes in the overlay. The expected coverage of the broadcast depending on packet loss ratio is the following:

$$B = (1+P)^d, \tag{4.1}$$

where P is the expected number of nodes getting the message for two immediate neighboring nodes and d is the length of the message path, which can be estimated as $\log_2 N$ (height of the broadcast tree), where N is the total number of nodes.

This can be easily inferred from the fact that the number of nodes getting the message from to immediate neighbors is $1+P$. The node that receives the message adds the expected value of $1*P$. Therefore, this leads us to get: $(1+P)+P(1+P)+P^2(1+P)...$ Which is summarized and generalized in the first expression.

By dividing the above expression (Eq. 4.1) by the number of nodes 2^d we get the ratio of nodes receiving the message m. The resulting expression after dividing can be seen above (Eq. 4.2):

$$m = \left(\frac{1+P}{2}\right)^d \tag{4.2}$$

In order to add redundancy to our previous equation, we will consider P as $1-P_l^{k_d}$, where P_l is the packet loss ratio and k_d is the level of redundancy. Substituting it into the previous expression, we get:

$$m = \left(\frac{2-P_l^{k_d}}{2}\right)^d \tag{4.3}$$

These equations were validated by their authors, getting even better results in the experimentation when using the redundant algorithm. Among other reasons, the principal cause was because when messages are duplicated, each node can receive the message from multiple paths, which can be shorter than the estimated by d.

4.7.2 Broadcast + SYNC

The broadcast by itself cannot guarantee that every TallyBox maintains a perfect membership of system, even with high redundancy without flooding the network with too much messages. For this reason, we propose a solution that uses less redundancy without (negligible) loss of robustness based on periodic reconciliation or anti-entropy.

More concretely, we call this solution SYNC phase and it consists on contacting periodically a number of random TallyBox servers and ask them for the last received updates. In this way, if a membership message is lost during the broadcast phase, it will be recovered from the nodes that received it without flooding the entire network with a new broadcast. The synchronization period can be adapted depending on the network activity and the desired refresh rate.

In this work we definitely assume that malicious TallyBoxes will be evenly distributed. This is because their IDs are assigned by the Authenticator server. Therefore, in our redundancy analysis, now we take into account not only the reliability of the broadcast, but also the reliability of combining the CAST with the SYNC phase.

Obviously, the SYNC phase will depend directly on the redundant broadcast, so we will also use the broadcast equation (Eq. 4.3) to get the result of our SYNC phase.

We must first analyze the situations in which the SYNC phase can fail, which are mainly two possible situations:

1. The first one is that the SYNC packet gets lost with probability P_l as we previously stated.

2. The second one is that the packet arrives to a node that will respond, but the contacted node does not have the message we want to synchronize with because it got lost when arriving the contacted node in the broadcast phase. So basically, the contacted node doesn't have the message we want.

Taking both situations into account, we defined the reliability S of the SYNC phase as:

$$S = 1 - (P_l + (1 - m)(1 - P_l)) \tag{4.4}$$

However, as explained above, sending a single SYNC message is clearly insufficient to re-synchronize correctly. Thus, redundancy, k_s, is added to the equation which yields:

$$S = 1 - (1 + m * (P_l - 1))^{k_s} \tag{4.5}$$

In summary, combining the CAST and SYNC phases, a message can be received in two ways in our proposal. The first way is through the broadcast phase, m. Or on the other hand, that happens when the message was not received, it will be obtained in the SYNC phase ($(1-m) * S$). So, the total system reliability M is given by the following equation:

$$M = m + (1 - m) * S \tag{4.6}$$

Chapter 5

Analysis

After proposing our novel method TallyNetworks, we are going to give both theoretical analysis and experimental study to verify its functioning. For the theoretical part, a security analysis is performed and discussed. Moreover, for the experimental part, we simulate a Broadcast + Sync over a network, we validate the protocol cryptographic operations and finally we evaluate the membership storage and cost.

5.1 Security Analysis

In Chapter 4 the four main security properties that our system should have were described. Now, we analyse how our proposed protocol TallyNetworks maintain those 4 security requirements:

- **Privacy**: As mentioned before, this property guarantees that the relation between a vote and the identity of the person who cast it cannot be discovered. In our protocol, we ensure this property thanks to the untraceability of the blind signature scheme and also the use of an anonymous bidirectional channel between participants and TallyBoxes. Our model only provides *pseudoanonimity* since the entire voting history of a given pseudonym is stored in the network. If it is the case that the participant is identified in any of the communications with a TallyBox, the whole voting history will be linked to his real identity.
- **Integrity**: Recall that this property guarantees that the result of the election cannot be altered in any way. In our particular system, the cases that have to be taken into account are:

    1. Allowing only registered users to cast votes.

2. Allowing users only to vote once
3. Making sure that votes are correctly tallied.

Below we explain how those properties are guaranteed for our protocol following the same order:

1. The first property is ensured thanks to the Authenticator's signature of the pseudonym. If the pseudonym is not signed, the vote will not be propagated into the network as it will not be considered as a valid vote.
2. The second property is ensured thanks to the user's signature of each vote. If a vote signed with an already used key is emitted, it will be discarded.
3. Finally, the third property is ensured thanks to the redundancy of TallyBoxes that cover each key. This allows TallyBoxes to wait for a minimum number of messages before considering the vote as valid. If two different votes signed with the same key are received at the same time, both are discarded.

- **Robustness**: As explained in last chapter, this property guarantees that the protocol is robust against external attacks or malicious nodes that try to disrupt the overall process. In our case in particular, our distributed overlay is designed with sufficient redundancy and communication to overcome such attacks. Evaluating our protocol, it is observed that the most vulnerable part of our architecture is the centralized Authenticator component. Namely, if it is the case that this component is not working due to attacks, the entrance of new nodes and users to the network is compromised. However, in any case, the distributed TallyNetwork can continue working independently from the Authenticator.
- **Verifiablity**: Analyzing the last requirement, we can advocate that our system provides individual verifiability. This is because a user can recover (via a GET request for a key) the votes of a poll from different TallyBoxes (key managers), and check if his vote is present and counted. Furthermore, user can check if all votes in a poll are correct according to their signatures and if the global count is consistent in the different TallyBoxes.

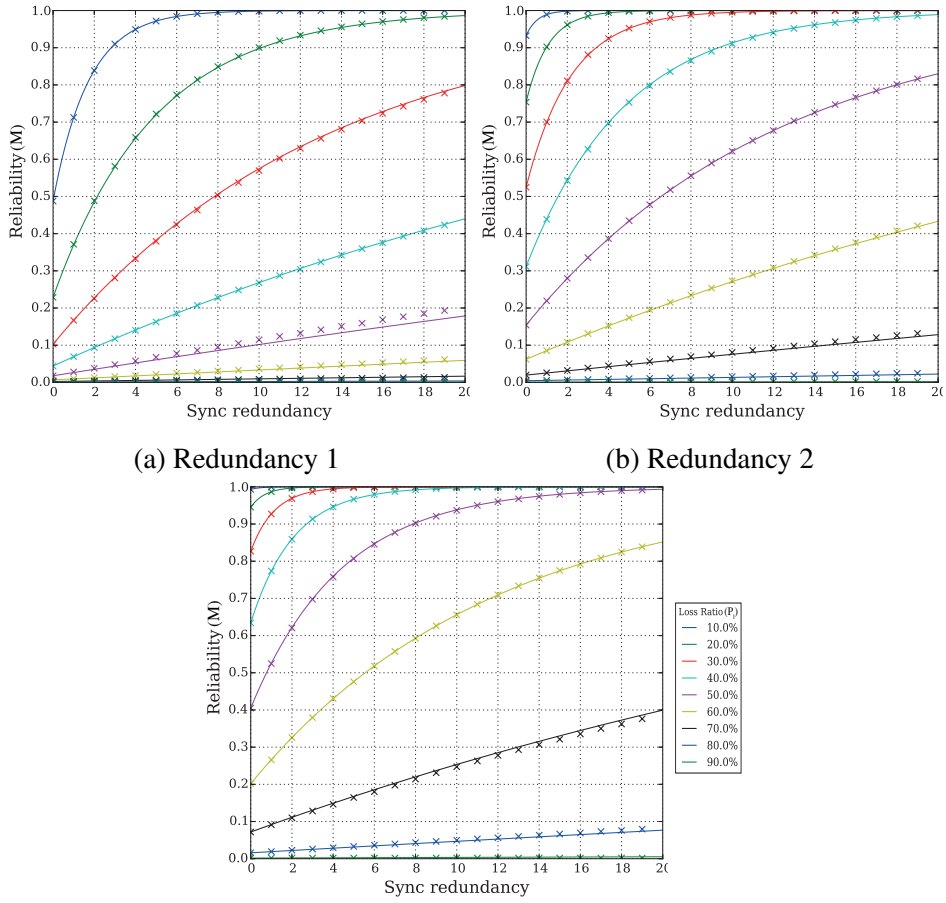
With this complete analysis of the four security requirements that we stated in last chapter, we can conclude that our system satisfies them.

5.2 Experimental Analysis

By means of simulations, whose properties are stated in each subsection, we evaluate 3 parts of our proposal. Firstly, we simulate the Broadcast + Sync to prove the improvement of the system due to the synchronization. Secondly, we validate the protocol cryptographic operations by evaluating their cost. Lastly, we evaluate the membership storage, showing that the space cost of storing the membership information is not a problem.

5.2.1 Broadcast + Sync Simulation

As it can be easily seen in Figure 5.1, synchronization improves the overall system reliability M. This is concluded according to the redundancy level determined by our equations. The analysis was done by simulating a network of $10,000$ TallyBoxes with Peersim [1].



(a) Redundancy 1

(b) Redundancy 2

(c) Redundancy 3

Fig. 5.1 Broadcast+Sync Evaluation

[1] For further information about this Peer-to-Peer Simulator visit: http://peersim.sourceforge.net

If we again observe Figure 5.1, it can be noted that the obtained results (denoted as crosses) faithfully follow our analytical model (represented as lines). However, we have to take into account that our evaluation is rather pessimistic. This is because our equations capture only the worst-case scenario, which is when all out-of-sync TallyBoxes try to re-sync at the same time, just after a broadcast. Nonetheless, if we consider a real situation, all nodes will not sync at the same time, which means that the probability of re-synchronizing increases after any other node has already synced. Therefore, we can conclude that a better reliability M will be achieved in practice.

Moreover, we want to point our that the initial reliability of the broadcast is the one that corresponds to the value 0 of the x axis (sync redundancy).

Analyzing Fig. 1(a), we can determine that the most relevant result obtained is that with $P_l = 10\%$ of loss ratio and no redundancy, reliability improves from 50% to 100% by just contacting 8 nodes. This is a relevant result since in previous works[9], in order to achieve the same result with just broadcasting, a minimum of 3 levels of redundancy would be required as shown in.

What is more, Fig. 1(b) illustrates that with redundancy 2, we only need to contact 2 nodes when $P_l = 10\%$ to achieve 100% reliability, and only 4 nodes to obtain 100% reliability with $P_l = 20\%$. Even 30% of losses can be overcome by simply contacting 10 nodes, improving from 50% to 100% the level of reliability. In order to achieve this with just broadcasting, at least a redundancy level of 4 would be required.

Finally, compared with the previous figure, Fig. 1(c) shows that the number of nodes required to ask for synchronization is smaller up to $P_l = 30\%$. Besides, it also shows that a reliability of 100% can be achieved with a loss ratio of 40% by only contacting 10 nodes. Furthermore, even a loss ratio of 50% could be supported by just contacting 20 nodes or even less.

Those results clearly show that our network is more tolerant to failures with less resources, thus saving significant amounts of bandwidth. Moreover, the syncing phase makes our network much less fragile in front any attack that wants to silence the public opinion, thanks to a our greater reliability.

For example, if redundancy 3 is used and 20 nodes are contacted when synchronizing, a coverage of 100% is assured for an up to $P_l = 50\%$ in a N size network. Therefore, in case that an attacker wants to break the broadcast of the network, she will need to introduce $N+1$ byzantine nodes. This has to be done as in order to generate a higher loss ratio than 50%. Recalling that loss ratio generated by byzantine nodes can be calculated as Byzantine Nodes/Total number of nodes, in our example it is $(N+1)/2*N$

Notice that both the broadcast redundancy and the syncing redundancy are not static parameters and can evolve with the network, i.e., increasing or decreasing their values according to the network state. Such state and its loss ratio can be estimated by the nodes themselves by calculating the ratio of their unanswered requests sent to other TallyBoxes. Thus, the algorithm can be adaptive to optimize system resources without compromising reliability.

5.2.2 Protocol cryptographic operations validation

In this part of the analysis we evaluate the cost of each protocol operation. In order to do so, we implemented each of them in Python using the Pycrypto library. To provide the evaluation we ran more than 10,000 tests in an Intel Core i5-3470@3.20GHz with Debian 7.8. As a result we obtained the numbers that are shown in Table 5.1.

Table 5.1 Protocol simulation

	Operations/second (depending on key size)		
Stage	1024	2048	4096
1. Participant Join			
1.1 P: Key generation	4,23	1,06	0,16
1.2 P: Public Key hashing	12500	10000	7142
1.3 P: Hash Blinding	20000	7142	2857
1.4 A: Blind Signature	657,89	110	18,16
1.5 P: Signature Unblinding	100000	49999	16666
2. TallyBox Join			
2.1 A: Credential signature	657,89	110	18,16
2.2 T: Signature verification	16666,66	5000	1538,46
3. Voting			
3.1 P: Vote signature	657,89	110	18,16
3.2 T: Vote signature verification	16666,66	5000	1538,46

P:Participant, A:Authenticator, T: TallyBox

As it can be observed form Table 5.1, the most limiting operations are signing and key generation. The latter is not a problem, since it is performed only once by each participant when joining. Signing is not a problem neither as it impossible that a real person will cast more than 657 votes per second. Consequently, the Authenticator is the only limiting entity. However, due to the fact that it is a trusted and controlled entity, it can be easily scaled to

perform much more operations per second. Furthermore, the Authenticator can rate limit the joining process to the network in order to reduce pronounced joining peaks.

5.2.3 Membership storage cost evaluation

The final part of our evaluation consists on evaluating the space cost of storing the membership information. The analysis is done assuming a 160-bit TallyBox server ID, an IP address of 32 bits, a port of 16 bits and a signature of variable size. A TallyBox stores this information for each TallyBox in the network. As it can be seen in the following results showed in Table 5.2 (in MB) and assuming 4096-bit signatures, it would take only around 513 MB of disk space. Therefore, we can confirm that space is not a problem.

Table 5.2 Storage Cost Evaluation (MB)

Network size	Signature Size (bits)		
	1024	2048	4096
10.000	1,47	2,69	5,13
100.000	14,69	26,89	51,31
1.000.000	146,87	268,94	513,08

Thus, we can claim that our combined SYNC & CAST algorithm is adaptive and resilient and can scale to big networks. Furthermore, thanks to full membership, we know the size of the network and depth of the tree, so that the aforementioned broadcast algorithm is efficient and feasible to implement in real networks.

Chapter 6

Conclusion and Future Work

In this work we have introduced TallyNetworks, an edge-centric distributed overlay for protecting the privacy of your on-line opinions. This model is aimed for typical user on-line participation tools such as open polls or item rating (`stars`, `like/dislike`).

Thanks to the decentralized edge-centric model, not only this method does protect your privacy, but also we have achieved that no one can have the entire control of the information, which makes people less vulnerable. This is because we have empowered people again, returning to them the control of their information. Furthermore, information will be more reliable as it cannot be manipulated by a centralized entity to impose an opinion and would make people much less tied to the current big players of the public opinions such as Facebook, Google or Amazon.

A TallyNetwork must both count opinions and also assure their correct retrieval under attacks or censorship attempts. Therefore, our proposed solution cover those problems. In this work, we showed how it is possible to provide privacy, integrity, robustness, and end-to-end verifiability through the combination of security technologies (blind signatures, encryption and anonymous channels) with a one-hop DHT of edge servers.

Due to the proof of concept that we implemented for this work, we could demonstrate that besides guaranteeing the above mentioned properties we can do it in a real environment with communities that have a huge number of members on the order of millions with quite quick response. It has been shown that we can process a large number of operations per second involved in the protocol.

Moreover, as our evaluation faithfully fits our theoretical model, nodes can estimate the optimal parameters in each moment, estimating the state of the network and its loss ratio by calculating the ratio of their unanswered requests sent to other TallyBoxes. Therefore, we can adjust the protocol parameters (broadcast and syncs) in order to evolve with the network,

increasing or decreasing their values according to the network state. Thus, the algorithm can be adaptive to optimize system resources without compromising reliability.

Although the good perspectives, it is noteworthy that our proposal also has a main drawback that we should take into account. We are moving the control of the information to the edges which a priori is a beneficial property that empowers people. However, we have recently been witness to how vulnerable we are in front of hacking attacks such as what happened with WannaCry attack [1] in which 200,000 computers were infected across 150 countries. If huge companies with high security requirements were affected, it is easy that ordinary people also gets infected. This means that TallyBoxes or even participant's smartphones are subject to the possibility of being hacked and then compromising user's privacy or even system's integrity if the scale of the attack is bigger than the expected loss ratio. Therefore, moving the power to the edges makes everything a bit more risky as they are more vulnerable.

Nevertheless, moving to a decentralized approach in which privacy is one of the main concerns or priorities seems to be a natural movement as everyday, people is more concerned about privacy. However, they are also concerned about the power that big players such as Google or Facebook are acquiring thanks to all of our opinions, votes, likes/dislikes... In addition, this shift towards a privacy-first service that can motivate decentralization cannot be only observed in our society but also in the new General Data Protection Regulation from the European Commission [2]. With this new document, the European Union is becoming much more strict in terms of privacy with companies and the data they use from European Citizens.

Furthermore, technology in terms of computation power on mobile phones and home devices along with connection bandwidth at home is improving everyday which makes possible and easy to build your own server with very high speeds both in terms of computation and connection, specially the ones powered by fiber optic. This makes much more feasible a decentralized model. In fact, recently we could see that some decentralized projects has been initiated with a remarkable adoption such as a decentralized social network called Diaspora [3] that has more than 667.000 users [46]. These positive numbers and a favorable scenario can be a great companion for our proposal as they demonstrate the feasibility and the interest that, at least in a certain big enough target, can awake a proposal like ours.

It is noteworthy that decentralization entails architectural and implementation challenges due to the nature of a decentralized system; (i) we cannot trust anybody, but we must

[1] For further information about the WannaCry attack visit: http://edition.cnn.com/2017/05/12/health/uk-nhs-cyber-attack/index.html or https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[2] For further information about the General Data Protection Regulation visit: http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf

[3] For further information about Diaspora visit: https://diasporafoundation.org

be sure that the system will finally provide the correct results for us. This means that we mainly need security and reliability (ii) we must coordinate hundreds, thousands or even millions of participants in our system, which means we need a very scalable system but also resistant to attacks, (iii) furthermore, in our case, everything must be achieved without compromising user's privacy. However, although the difficulty of overcoming such challenges, our proposal has demonstrated that it can provide privacy in a decentralized system thanks to the combination of pseudonymous with blinds signatures and a one-hop structured overlay based on Kademlia. Such overlay provides us not only scalability, but also, combined with our protocol based on broadcast and syncing phases, provides integrity, end-to-end verifiability and robustness to malicious attacks.

We want to point out that this work is the first steps towards the construction of an edge-centric middleware based on an efficient and adaptive one-hop structured overlay of stable edge servers.

As a relevant remark, we started our path to bring privacy-first public opinions systems and moving away from this post-privacy world leaded by big players instead of persons, presenting our TallyNetworks proposal in the 22nd International European on Parallel and Distributed Computing Conference (Euro-par). In fact, during the conference we obtained a positive feedback that encourages us to keep working on TallyNetworks and continue walking towards our goal.

What is more, the next steps we would like to reach consist on bringing to our proof of concept the necessary strength to be able to use it in a real life environment. Additionally, we want to prepare an application that uses our system as a use case with a real community of users in order to be able to study how the system reacts in a real environment.

Once we reach a mature implementation of our proposal, we would like to study the possibility of modifying and adapting a mobile application called ConsultApp that has been developed in the Architectures and Telematic Services (AST) Research Group in the Rovira i Virgili University. ConsultApp is a direct democracy system that permits to create open polls by any member of the community. The users can vote or comment on the arguments for or against each option of the polls. For this reason, ConsultApp would be a very good use case to use TallyNetworks. However, it is a challenging task as it was developed for Android devices as an API consumer application of a centralized service that stores all the votes. The proposal would consist on substituting such centralized service with a TallyNetwork as backend.

Moreover, we would like to use the resulting TallyNetwork application in our University to favor the global discussion and promote internal democracy. Doing so, we could demonstrate that privacy-first systems that empower people again are possible in real life scenarios.

However, as it can be seen, this a very ambitious proposal that would require huge development effort. Therefore, we can consider this as a mid-long term goal that requires further study. Meanwhile, we will work improving our proposed method and we will try to reach a more mature implementation to be able to consider goals such as the one above mentioned.

The main improvement we would like to work on is to enact measures to prevent impunity associated to complete anonymity. For example, identity disclosure if a user is trying to tamper the results with invalid votes or voting two times for different options. We believe that this tool can promote democracy while respecting the privacy of the opinions of the members of any community. Moreover, enacting such measures would improve the democracy quality, but without affecting user's privacy.

In summary, we have presented TallyNetworks, an edge-centric distributed overlay for protecting the privacy of your on-line opinions providing privacy, integrity, robustness, and end-to-end verifiability through the combination of security technologies with a one-hop DHT of edge servers. This is our contribution with the aim of moving and pushing the world from a post-privacy to a privacy-first era.

References

[1] Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R. (2003). P-grid: a self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3):29–33.

[2] Cerveró Abelló, M., Mateu Meseguer, V., Miret Biosca, J. M., Sebé Feixas, F., and Valera Martín, J. (2014). An elliptic curve based homomorphic remote voting system.

[3] Chan, P. H., Li, H., Ugalde, J., and Stoller, Y. (2015). Private decentralized e-voting.

[4] Chaum, D. (1983). Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer.

[5] Chaum, D. (1985). Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044.

[6] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90.

[7] Cohen, J. D. and Fischer, M. J. (1985). *A robust and verifiable cryptographically secure election scheme*. Yale University. Department of Computer Science.

[8] Czirkos, Z. and Hosszú, G. (2012). Enhancing the kademlia p2p network. *Periodica Polytechnica Electrical Engineering*, 54(3-4):87–92.

[9] Czirkos, Z. and Hosszú, G. (2013). Solution for the broadcasting in the kademlia peer-to-peer overlay. *Computer Networks*, 57(8):1853–1862.

[10] Danezis, G. and Diaz, C. (2008). A survey of anonymous communication channels. Technical report, Technical Report MSR-TR-2008-35, Microsoft Research.

[11] Davidovic, V., Ilijevic, D., Luk, V., and Pogarcic, I. (2015). Private cloud computing and delegation of control. *Procedia Engineering*, 100:196–205.

[12] Davis, H. (2006). *Google advertising tools: Cashing in with AdSense, AdWords, and the Google APIs*. " O'Reilly Media, Inc.".

[13] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. Technical report, DTIC Document.

[14] Dingledine, R., Mathewson, N., and Syverson, P. (2008). Tor: anonymity online.

[15] Douceur, J. R. (2002). The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer.

[16] Fujioka, A., Okamoto, T., and Ohta, K. (1993). A practical secret voting scheme for large scale elections. In *Advances in Cryptology—AUSCRYPT'92*, pages 244–251. Springer.

[17] Ganti, R. K., Ye, F., and Lei, H. (2011). Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11).

[18] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., and Riviere, E. (2015). Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42.

[19] Gupta, A., Liskov, B., Rodrigues, R., et al. (2003). One hop lookups for peer-to-peer overlays. In *HotOS*, pages 7–12.

[20] Hendrikx, F., Bubendorfer, K., and Chard, R. (2015). Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184–197.

[21] Huszti, A. (2011). A homomorphic encryption-based secure electronic voting scheme. *Publ. Math. Debrecen*, 79(3-4):479–496.

[22] Jakobsson, M. (1998). A practical mix. *Advances in Cryptology—EUROCRYPT'98*, pages 448–461.

[23] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM.

[24] Kinateder, M. and Rothermel, K. (2003). Architecture and algorithms for a distributed reputation system. *Trust Management*, pages 1071–1071.

[25] Li, R. (2009). Distributed hash table.

[26] Li, W., Santos, I., Delicato, F. C., Pires, P. F., Pirmez, L., Wei, W., Song, H., Zomaya, A., and Khan, S. (2016). System modelling and performance evaluation of a three-tier cloud of things. *Future Generation Computer Systems*.

[27] Marc O'Morain, V. T. and Verbuggen, W. (2016). Onion routing for anonymous communications. [Online; accessed 4-June-2017].

[28] Mateu, V., Miret, J. M., and Sebé, F. (2011). Verifiable encrypted redundancy for mix-type remote electronic voting. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 370–385. Springer.

[29] Maymounkov, P. and Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer.

[30] McDowell, M. (2004). Understanding denial-of-service attacks. *National Cyber Alert System, Cyber Security Tip ST04-015.2004*.

[31] Middelesch, E. (2015). Anonymous and hidden communication channels: a perspective on future developments. Master's thesis, University of Twente.

[32] Montresor, A. (2016). Reflecting on the past, preparing for the future: From peer-to-peer to edge-centric computing. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 22–23. IEEE.

[33] Peng, K. (2011). An efficient shuffling based evoting scheme. *Journal of Systems and Software*, 84(6):906–922.

[34] Peng, K., Aditya, R., Boyd, C., Dawson, E., and Lee, B. (2005). Multiplicative homomorphic e-voting. *Progress in Cryptology-INDOCRYPT 2004*, pages 1403–1418.

[35] Peris, A. D., Hernández, J. M., and Huedo, E. (2016). Evaluation of alternatives for the broadcast operation in kademlia under churn. *Peer-to-Peer Networking and Applications*, 9(2):313–327.

[36] Ritter, J., Haenni, R., and Neumann, S. (2014). *Decentralized e-voting on android devices using homomorphic tallying*. PhD thesis, Master Thesis, Bern University of Applied Sciences, Biel, Switzerland.

[37] Rodríguez, M. R., López, P. G., and Sánchez-Artigas, M. (2017). *TallyNetworks: Protecting Your Private Opinions with Edge-Centric Computing*, pages 211–223. Springer International Publishing, Cham.

[38] Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer.

[39] Sako, K. and Kilian, J. (1995). Receipt-free mix-type voting scheme. In *Advances in Cryptology—EUROCRYPT'95*, pages 393–403. Springer.

[40] Sànchez-Artigas, M., Pujol-Ahulló, J., Pamies-Juarez, L., and García-López, P. (2010). p2pweb: An open, decentralized infrastructure of web servers for sharing ephemeral web content. *Computer Networks*, 54(12):1968–1985.

[41] Schneier, B. (1996). Applied cryptography john wiley. *New York*.

[42] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160.

[43] Weldemariam, K., Volkamer, M., and Villafiorita, A. (2011). Evoting: What we learned where we are going to. In *Proceedings of the Sixth International Workshop on Frontiers in Availability, Reliability and Security,(FARES 2011)*.

[44] Wikipedia (2016). Mix network. [Online; accessed 4-June-2017].

[45] Wikipedia (2017a). Blind signature. [Online; accessed 18-May-2017].

[46] Wikipedia (2017b). Diaspora (social network). [Online; accessed 3-June-2017].

[47] Wikipedia (2017c). Distributed hash table. [Online; accessed 3-June-2017].

[48] Wikipedia (2017d). Electronic voting. [Online; accessed 4-June-2017].

[49] Wikipedia (2017e). Zero-knowledge proof. [Online; accessed 6-June-2017].

[50] Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on Knowledge and Data Engineering*, 16(7):843–857.