POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Stanisław Barański
Nr albumu: 160518
Studia drugiego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Specjalność: Sieci komputerowe

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Wykorzystanie technologii blockchain i procesów infekcji w grafach do obrony przed atakami zatruwania treści w sieciach informacjocentrycznych

Tytuł pracy w języku angielskim: Application of blockchain and infection processes in graphs to mitigate content poisoning attacks in information-centric networks

Potwierdzenie przyjęcia pracy	
Opiekun pracy *podpis*	Kierownik Katedry/Zakładu (pozostawić właściwe) *podpis*
dr hab. inż. Jerzy Konorski	

Data oddania pracy do dziekanatu:

GDAŃSK UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRONICS, TELECOMMUNICATIONS AND INFORMATICS

DECLARATION regarding the diploma thesis titled:
Application of blockchain and infection processes in graphs to mitigate content poisoning attacks in information-centric networks

First name and surname of student: Stanisław Barański
Date and place of birth: 18.03.1995, Gdańsk
ID: 160518

Faculty: Faculty of Electronics,Telecommunications and Informatics
Field of study: informatics

Cycle of studies: postgraduate
Mode of study: Full-time studies

Aware of criminal liability for violations of the Act of 4th February 1994 on Copyright and Related Rights (Journal of Laws 2018, item 1191 with later amendments) and disciplinary actions set out in the Act of 20th July 2018 on the Law on Higher Education and Science (Journal of Laws 2018 item 1668 with later amendments),[1] as well as civil liability, I declare that the submitted diploma thesis is my own work.

This diploma thesis has never before been the basis of an official procedure associated with the awarding of a professional title.

All the information contained in the above diploma thesis which is derived from written and electronic sources is documented in a list of relevant literature in accordance with art. 34 of the Copyright and Related Rights Act.

I confirm that this diploma thesis is identical to the attached electronic version.

Gdańsk, ...
signature of the student

[1] The Act of 20th July 2018 on the Law on Higher Education and Science:

Art. 312, section 3. Should a student be suspected of committing an act referred to in Art. 287 section 2 items 1–5, the rector shall forthwith order an enquiry.

Art. 312, section 4. If the evidence collected during an enquiry confirms that the act referred to in section 5 has been committed, the rector shall suspend the procedure for the awarding of a professional title pending a judgement of the disciplinary committee and submit formal notification on suspicion of committing a criminal offence.

STRESZCZENIE

Sieci informacjo-centryczne (ang. Information-centric Networking (ICN))—które kandydują do miana sieci przyszłości—zmagają się z nowym wektorem ataków. Jednym z ataków jest atak zatruwania treści (ang. Content Posioning Attack (CPA)), a zwłaszcza jego silniejsza forma—atak zatruwania treści przez fałszywe dane (ang. Fake Data CPA). Sieci te, są wyjątkowo podatne na ten typ ataków, ponieważ nie pozwalają na usuwanie treści—w momencie, gdy treść trafia do sieci, nie jest możliwe jej usunięcie inaczej niż poprzez wygaśnięcie. Aktualnie znane metody uwierzytelniania jak login/hasło, klucze prywatne, biometria, potwierdzenie SMS/email operują w jednym tym samym wymiarze uwierzytelniania. W tej pracy proponujemy nowy wymiar uwierzytelniania, który bazuje na dostępnie do czasu. Kiedy napastnik-wydawca operuje w czasowo ograniczonym środowisku, jego dostęp do skradzionych uwierzytelnień jest czasowo ograniczony, podczas gdy prawdziwy wydawca nie jest ograniczony czasowo w żaden sposób. Ta różnica jest wykorzystana w celu stworzenia nowego systemu uwierzytelniania. Dwie implementacje systemu są zaproponowane, pierwsza bazuje na algorytmie infekcji na grafach, a druga na łańcuchu bloków (ang. Blockchain). Dzięki zbudowanym symulatorom jesteśmy w stanie zaobserwować problem Sojuszu Obronnego (ang. Deffensive Alliance) w algorytmie infekcji na grafach. Obie metody zostają ocenione pod kątem złożoności komunikacyjnej, determinizmowi algorytmów, możliwości dopasowania się do zmian, oraz odporności na błędy (również silniejszej formy bizantyjskich generałów).

ABSTRACT

Information-centric networks—that are candidates for the title of the network of the future—struggle with a new vector of attacks. One of them is content poisoning, especially the stronger form—Fake Data CPA. ICN networks are especially prone to those types of attacks due to a lack of removal functionality. Currently, known authentication methods like login/password, private key, biometry, SMS/email confirmation operate on the same dimension of authentication. We propose a new authentication dimension, which is time availability. When an adversary publisher is operating in a time-constrained environment, his access to target credentials is limited, whereas honest publisher is not timely constrained in any way. We leverage such distinction to propose a new authentication mechanism. Two implementations are proposed, the first one based on infection processes in graphs, and the second one based on blockchain technology. We provide simulators that give us interesting observations, such as the Defensive Alliance problem in graph infection algorithm. We evaluate both approaches in terms of communication complexity, scalability, determinism, resilience, and fault-tolerance (including its stronger form—Byzantine-fault).

CONTENTS

LIST OF IMPORTANT SYMBOLS AND ABBREVIATIONS

NDO – Named Data Object, an object primitive used in NDN networks.

ICN – Information-centric network.

NDN – Named Data Networking.

CPA – Content Poisoning Attack.

Fake Data CPA – A stronger form of Content Poisoning Attack where the adversary has access to credentials and can forge valid signature on poisoned data.

WoT – Web of trust graph generator.

PB – Probabilistic duplication graph generator.

RNG – Random graph generator.

Proof-of-Time – Proof of access to credentials over some period.

Credibility Score – Number of collected proof-of-time claims over total required number of proof-of-times.

GI – Graph Infection algorithm [1].

FBA – Federated Byzantine Agreement consensus protocol [2].

SCP – Stellar Consensus Protocol, the construction of FBA.

1. INTRODUCTION

Current Internet architecture based mostly on TCP/IP stack allows establishing communication channels between two IP addresses. While this model worked for past years, now it struggles to fit current demands. Today the Internet is dominated by transporting content such as audio, video, images, and text from content creators to content consumers. Information-centric networks introduce a paradigm shift from a host-oriented to a content-oriented paradigm. By placing content in the center of interest, it allows for achieving several benefits. All network participants become more aware of transferring content. This kind of awareness allows implementing various improvements over host-based paradigms such as content caching, mobility, integrity, and security assurance. All of those features are guaranteed natively by an ICN transport layer.

Many projects implement the ICN approach: Data-Oriented Network Architecture (DONA), Named Data Networking (NDN), Publish-Subscribe Network Technology (PURSUIT), Scalable and Adaptable Network Solutions (SAIL), Inter-planetary File System (IPFS).

They all differ in details, but the core concepts are the same, they try to achieve a pull-driven communication model suited for disconnections, disruptions, mobility, and transferring large data to many devices. They introduce in-network storage on each node for data caching. Also, decouple senders from receivers by resolving content by its name—not a host's location. In ICN, a data piece is requested by its name (Named Data Object - NDO) and is served by the closest possible node that is storing it, allowing efficient caching on the transport layer—relieving an application layer from implementing a cache strategy individually.

When we write NDO, we understand any arbitrary data piece that can be transferred over the network: a web page, music, images, video, documents, and data streams. Most importantly, NDOs, compared to traditional URLs, are location independent. They do not specify the location where the content should be served from, nor how it should be transferred to the receiver. NDO granularity may differ from packets to data chunks to whole data objects, depending on the approach and data size.

Data naming is the most significant concept in ICN. Data names must be unique—similarly to hostnames in current Internet architecture; two different DNS nodes must resolve one domain name to the same location—two different ICN nodes must resolve one name to the same content. There are two approaches to a naming system: 1) hierarchical - similar to URL, and 2) flat - global namespace, often just a hash of the file. Additionally, each NDO should be authenticated by the publisher who created it. Digital signatures on data objects guarantee both integrity and authentication. If a data piece is trusted, not the host, then it can be served from any untrusted source—and we still can trust it.

The most significant benefit of ICN networks is content caching. It turns out that this benefit becomes a double-edged sword when we take into account security concerns. If everything that makes a content authentic is a valid signature, an intruder who gets access to the publisher's private key can create poisoned content with valid signatures. Such content can be easily disseminated to ICN nodes, but hardly removed due to the lack of removal operation in ICN architecture.

This thesis aims to provide a mechanism to protect subscribers from trusting poisoned

content, even in situations when an intruder has access to a stolen private key (making it indistinguishable from the actual publisher).

This thesis contributes to the field of ICN networking by

1. Formulating the general approach to protect a subscriber from a content poisoning attack.
2. Building simulators that allow analyzing a graph infection implementation proposed by [1].
3. Observing a Defensive Alliance phenomenon that causes problems in the GI algorithm.
4. Proposing a new implementation based on Blockchain technology.
5. Evaluating and comparing those two approaches in the context of scalability, complexity, fault-tolerance, and resiliency.

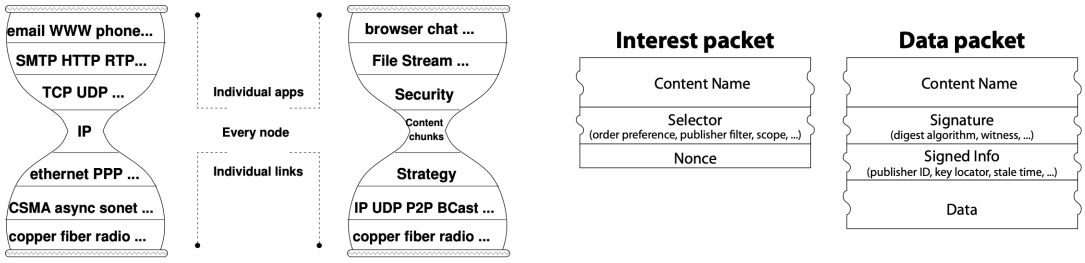
The remainder of this thesis is organized as follows. In Sec. 1.1, we present an example project implementing ICN architecture; we decided on Named Data Networking (NDN) since it is the most popular one. In Sec. 2, we describe a problem of Content Poisoning Attacks and in Sec. 3 and 4 an approach to solve it. In Sec. 5 we study epidemiology models to get intuition to graph infection algorithm proposed in [1] and described in Sec. 8. Then in Sec. 8.1, we describe simulators used to analyze the algorithm and in Sec. 8.2 observations that show weaknesses of such an algorithm. In Sec. 9, we generalize our initial problem to a consensus problem, which we then solve by using Blockchain technology and Federated Byzantine Agreement consensus protocol (described in Chapter 10). Finally, a comparison of both protocols is presented in Sec. 11.

1.1. Named Data Networking

To get a better understanding of how ICN networks work, we describe one of the most mature projects. The design was initially proposed under the name Content-Centric Network by Van Jacobson in 2006 [3]. Currently, the project is under development with the name Named Data Networking (NDN) [4].

NDN replace TCP/IP protocol stack by placing chunks of named content in place of IP addresses (see Fig 1.1a). The only layer that cannot be changed is the layer 3—the thin ”waist” of the stack. All protocols in the rest of the layers can be adjusted to the needs. NDN defines two types of packets: Interest, and Data (see Fig. 1.1b). An Interest packet, as the name suggests, indicate a interest of some data chunk, where a data chunk is identified by its Content Name. A consumer broadcasts an Interest packet to all its connection faces. If any node that receives such a message has the corresponding data locally, it can respond with a Data Packet, otherwise the Interest is forwarded to next router. In Fig. 1.2 we can see the forward engine model in NDN node. When a request arrives at one of available faces it gets dispatched based on the lookup result.

- FIB (Forwarding Information Base) is responsible for forwarding the interests to potential nodes where the data can be found (similar to FIB used in IP, except it allows for multiple outgoing faces rather than a single one).



(a) (b)

Figure 1.1. Named Data Networking design. Source [5]

- Content Store(CS) is responsible for buffering content requested by the Interest packet. Since the data can be served to multiple consumers——not just a single connection, as it is done in IP—it is not recycled after the first Interest request is satisfied. This mechanism is called in-network storage because the network itself is storing the data. It is possible due to idempotency, self-identification, and self-authentication of packets, so each packet is equally useful for many consumers (e.g., two hosts interesting in Netflix video can receive content from a single node without requesting the Netflix servers). Data is stored as long as possible, and different storage policies can be used, e.g., LRU replacement.
- PIT (Pending Interest Table) stores information about the interests forwarded to potential data sources. When one of them replies with a Data packet, the PIT is checked if some consumer is waiting for the data, and if so, the Data packet is forwarded to corresponding face.

To better understand the model, a concrete packet flow is presented in Fig.1.3.

1. A requester application wants to receive some data. It sends a request to its local NDN engine. The node first checks the local CS, and since the content is absent, it requests the Interest packet to the closes router.

2. The router first checks its Content Store, and if the data is absent, it checks if there is already a pending interest in the PIT for such data. If not, the request is forwarded to the next router, according to FIB.

3. The next router does the same as the previous router. It checks its CS, PIT, and then forward the Interest packet through FIB.

4. Finally, the Interest reaches the data source router, where the content is present in the CS, so it is immediately returned.

5. The router receives the Data packet and stores it in its CS, check which nodes are waiting

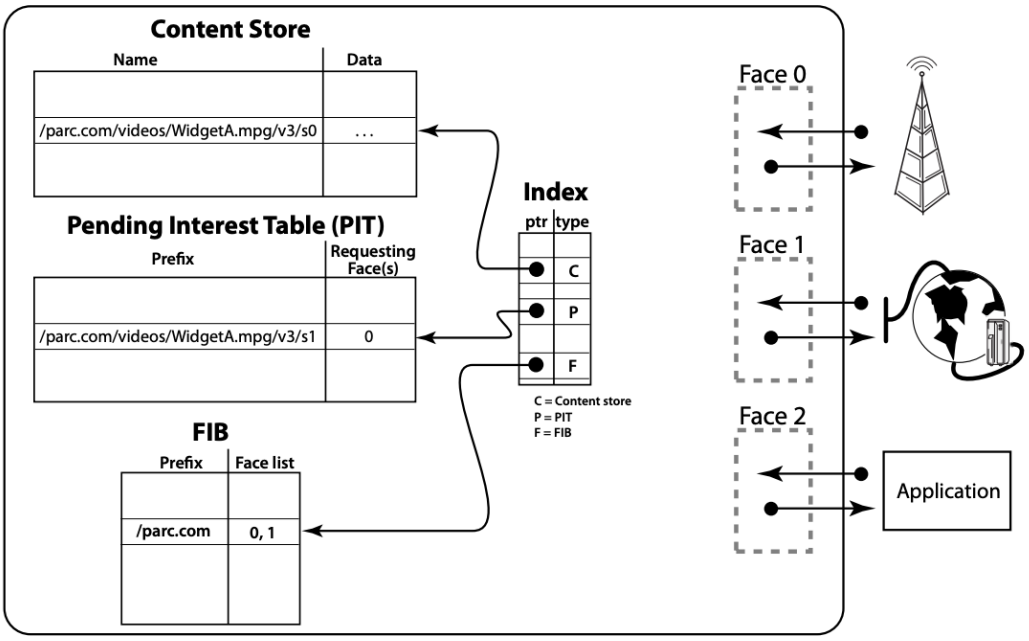


Figure 1.2. NDN node operations. Source [5]

for the data in PIT and forwards the Data packet accordingly.

6. This router (initially requested by the requestor) does the same as the previous router, storing the data in CS and responding with the Data packet.

7. After some time, the different requester sends the Interest packet to its closest router.

8. Router looks up the Content Name and finds out that the content is already stored in CS. Therefore it immediately returns the Data packet to save the network bandwidth and reducing the latency.

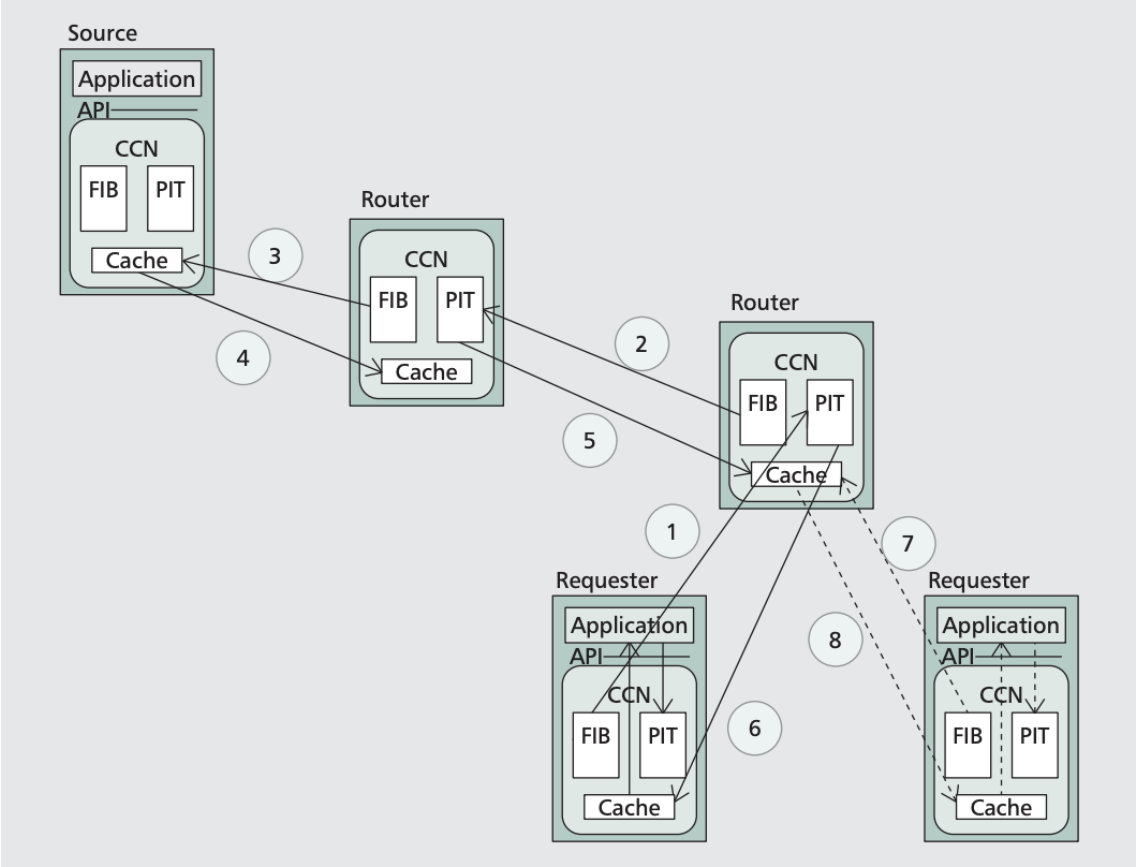


Figure 1.3. NDN packets flow. Source [6]

2. CONTENT POISONING

First, we narrow the area of research in this thesis. Content Poisoning in NDN (and possibly in all ICN networks) can be divided into two categories: 1. Corrupted Data, and 2. Fake Data. In both of them, content gets modified. The first one assumes that an attacker does not possess a signing-key of an authentic publisher; thus, the content will not pass a signature verification process. It might look like the problem is resolved. Unfortunately, signature verification is an optional process, and there are two reasons why this process might not be done on every router. The first one is economic incentivization. The process of signature verification is resource-consuming; thus, businesses might decide to skip it. The other reason is that a signature verification requires a corresponding public key. We cannot expect from router to store all possible public keys. Thus the routers are vulnerable to Corrupted Data CPA. This kind of attacks has been addressed in papers [7] [8] [9].

The second CPA category—Fake Data—is more dangerous because it assumes that an attacker gets access to the publisher's private key. Therefore he can forge a valid signature, which will successfully pass verification on an end-system.

Once a bogus content gets propagated across multiple nodes (more specifically on their Content Stores), each end-user who requests content from that node receives the poisoned content. Signature verification passes successfully; hence a user is convinced that the content is authentic. Moreover, the content cannot be as easily revoked as it was disseminated because most ICN networks do not allow removing/revoking its cache content, other than natural (e.g., LRU based) cache aging. The one way to effectively prevent fetching fake content is by explicitly filtering the name of the content we consider poisoned. However, this raises another problem, how a user can know the name of the poisoned content, and if it is not too late. In [9], we can read, "Therefore, it (Fake Data CPA) is harder to create but impossible to detect by end-systems." At the time of writing, the problem is still unsolved, but there is one proposal [1] which we research, expand, and modify. Before that, we will briefly review how other content-oriented services solve the Fake Data CPA problem.

2.1. Wikipedia

Wikipedia is an example of a system that faces a content poisoning problem. Let us investigate how do they solve the problem.

"Wikipedia is a free online encyclopedia, created and edited by volunteers around the world and hosted by the Wikimedia Foundation." Everyone can become a Wikipedia editor just by creating a free account. Account registration does not even require an email address. An editor can create a new article or modify an existing one. Before a change is publicly available, other editors read the proposition and have a chance to reject it or propose a further edit. If they cannot achieve consensus, they start a discussion and exchange their arguments. Finally, if they agree on one version, a consensus is achieved, and the new version of an article is published (keeping the whole history of changes). Such ease of account creation makes Wikipedia vulnerable to Sockpuppeting (also known as Sybil attack). This vulnerability allows one user to create multiple

accounts with different identities. Consequently, one person controls fake ”public opinion,” which can support his position in edits discussions. That is why a raw voting system is a non-preferred method in conflict resolutions. Wikipedia fights content poisoning attacks by using humans to detect bogus changes. This mechanism seems to work well in such a service, but it does not scale to a public Internet protocol, so cannot be used as is.

2.2. LOCKSS

LOCKSS is a decentralized p2p digital preservation system for libraries, developed at the Standford University [10]. Currently, it is an open-source, decentralized system used by many institutions. It is helping them to maintain a digital collection of journals, articles, and books. If we abstract from a specific type of data the system is operating on, we notice some similarities with ICN networks that we believe are worth investigating. The main difference between LOCKSS and ICN is that the former is very conservative; it rather prevents than expedites change of data. Nevertheless, this property may be useful in the context of preventing bogus data diffusion. In LOCKSS, each participating library becomes a node in a p2p network. It runs software responsible for collecting new content from e-journal websites by a crawler, serving already stored materials to local readers, and cooperating with other nodes in preserving materials when they get damaged. Due to publishers' copyrights, no node has to redistribute its content blindly to other peers. Other nodes can help repair damaged or completely missed material only to nodes that previously proved its ownership. New materials can only be acquired directly from a publisher's website to libraries which paid for subscription. Cyclic polls detect damages. Each interested peer votes on a hash of an Archive Unit (AU), the smallest unit in which nodes identify content. Since each node consists of a different set of Archive Units, the protocol treats each AU independently. If it turns out that some node contains AU with a different hash than most of the poll, it starts a sequence of repairs. As a result, LOCKSS becomes a self-healing store of data and does not increase the risk of free-loading non-purchased content. LOCKSS is designed in a way that does not rely on long-time public-key cryptography (a system that must operate for decades is eventually highly susceptible to private-key leakage). Since it does not rely on public-key cryptography, peer identity management is minimalist; therefore, such a system cannot rely on peer reputations.

2.3. Social Media and Fake News

Fake News is deliberate disinformative news that is hard to identify and can lead to destructive consequences if not mitigated in time. Social media are a perfect ecosystem for disseminating fake news; specially designed algorithms serve us content that we are likely to agree with. In [11], the authors point out why social media algorithms are very effective in fake news dissemination. Social media allow us to form like-minded people into groups more accessible than ever before. Such groups facilitate the Echo Chamber Effect, where each individual is surrounded by people who share and produce content that fits our current worldview. It leads to a segmented and polarized society, which is very prone to believing fake news that confirms current opinions, even if there is limited or no reason to believe it.

Fake news is often written in a very provocative fashion, making such content more attractive for interaction; more interaction leads to higher dissemination, fostering even more interaction. While for honest publishers and valuable content, this feature is helpful, it becomes a hazardous tool in the hands of malicious publishers. Additionally, the low cost of creating new accounts makes it even easier for an attacker to initiate the snowball effect, using fake accounts. Bots can interact with people or even with other bots, creating fake social opinions.

We notice that this model fits perfectly into our Content Poisoning Attack model. In some sense, we are facing a similar problem fake news does in social media: we want to allow valuable content to spread as fast as possible, while limiting malicious content dissemination to a minimum.

2.3.1. Mitigating fake news

One idea is to create centralized services that reveal known fake news. Each time someone is susceptible to some news, they can check if the news is not present in blacklisted news. This solution has some flaws. Attackers can publish honest news in such an ”oracle” service, leading to revoking genuine news, which can be as harmful as publishing fake news. A better way would be to check many independent ”oracle” services and arrive at a decision whether to believe based on how many of them are skeptical of such news.

Another idea would be to create certification services. Each news could be signed by services that decide which content is authentic. A user could decide which certification services he/she trusts, and therefore inherit the trust in the news signed by such services. This idea seems to have worked for decades in journalism. A reliable journal reader does not need to check each article if it is fake or not. He inherits each article’s confidence because he trusts that the journal’s reviewers have eliminated fake or low-quality content. Additionally, a journal publisher has no interest in publishing low-quality content because he/she has economic incentives to become as creditworthy a publisher as possible. This idea has some downsides. The process of reviewing an article is prolonged and requires qualified human interaction. Although some of the work can be outsourced to artificial intelligence, it is still a complicated task. Additionally, a journal publisher can introduce censorship or favor some kind of content. In other words, this mechanism does not scale for general-purpose content certification.

Let us consider fake news detection in practice. If Alice forgets to log out from social media account on a public library computer, someone can submit content, a post, that says, ”I do not want to live anymore the way everyone else is living, this rat race is not for me, from today I become homeless. Goodbye.” Such content is an excellent candidate to become successful fake news mainly because it is created from an author’s account, which gives it high credibility, but also because it contains some arguable truth that can make it trustworthy. Unfortunately, as we discussed in the previous certification services approach, this idea of content evaluation does not scale to the level we need. We notice that to decide if a post is fake or not, we can just wait. In this situation, the time is playing a crucial role, because if the post is deleted from Alice’s account after a short period, we can be sure that Alice would never have posted it. On the other hand, if the post is present for a long time on her page, we can state with high probability that she wants

it to be there, thus it is authentic content.

This approach also has its downsides, namely, it is slow. If a piece of information is urgent, like information that some country has started World War III, we do not have hours to wait until the information gets deleted or not. We need to act fast. In such situations, this approach is inefficient. But for less urgent content, this approach offers several useful features. It scales well–for each content publication, only one person is involved to remove the content (in the case when it is bogus). It is maintenance-free–by default, no one needs to do anything if the content is authentic. It is flexible–a reader can individually decide if the time from publication is long enough to trust such content or not. If the information is not very impactful, we can trust it from the first hour from the publication; otherwise, e.g., in the case of a banking webpage, we could wait a few hours until we enter the credentials into it. It is like HTTPS and HTTP, but much more flexible–some pages can be safely browsed over HTTP, while others should be used only over HTTPS.

As discussed before, the fake news problem in a social network is very similar to the CPA problem in ICN networks. So the solutions that work in one domain could work in the other. In this thesis, we will continue the idea of a content time availability, which we will call *Proof-of-Time*.

3. PROOF OF TIME

In the previous chapter, we introduced a mechanism for mitigating fake news propagation on social media accounts based on waiting—in the hope that bogus content will finally be removed, distinguishing it from authentic one. In this chapter, we generalize the mechanism to solve the CPA problem in ICN networks. Before that, we make some assumptions. First of all, an attacker is operating in a time constrained environment. He/she eventually loses access to stolen credentials, either by revoking, expiring, or any other external factor. Additionally, he/she is unable to refresh the credentials infinitely long. As a result, his/her access (measured in time) to a publisher's account is limited, in contrast with real account owners whose access is unrestricted. We leverage the difference between those two scenarios to create a new method of authentication. This method requires each publisher to prove access to credentials over some period of time, hence the name *Proof of Time*.

If a traditional model of authentication is based on proving access to credentials associated with an identity, then we can write

$$access\ to\ credentials \implies authenticity$$

. In this thesis, we extend this model to access to credentials over some period of time. We write

$$access\ to\ credentials \wedge access\ to\ time \implies authenticity$$

In the previous section, we discussed how the mechanism could work in social media networks. ICN networks are different in that they do not allow to remove content because it is distributed across many routers within their data cache (Content Store), to ease content dissemination. To support content removal, each node would need to periodically ask a publisher if the content has been removed or not, in In this way, sacrificing scalability, which was the reason the ICN was created in the first place. Instead of requiring a publisher to remove content, we create an overlay network of trust, where trust in content serves as a primitive. When the trust is low, a user should be discouraged from using the content (similar to how modern web browsers discourage usage of the insecure HTTP protocol).

In such an overlay trust network, each piece of content has its Credibility Score. The Credibility Score is gained by proving access to credentials over some period of time. In other words, the longer a publisher is able to sign content, the higher is the Credibility Score for the content. With that model, we can assume that a malicious actor is unable to achieve enough Credibility Score to convince a network about its trustworthiness, while a legitimate publisher will eventually achieve it over time. Later on we will discuss how to design such an overlay network in order to accomplish such an authentication mechanism.

4. NETWORK STRUCTURE

In the previous chapter, we mentioned the need to create an overlay network of trust where each publisher tries to convince the rest of the network about its trustworthiness by increasing the Credibility Score of the published content. In this chapter, we will discuss the considered network in more detail.

4.1. Trust propagation

As discussed before, the Credibility Score will spread across the ICN network nodes. It can spread in various ways, and in this chapter we will explore different models for this kind of mechanism.

One proposal[1] abstracts the concept of trust to the concept of infection and so allow us to adopt the knowledge created by biological research. In our case, each node that develops trust in some content is called infected. Each node that does not trust the content is called healthy. By default, all nodes are healthy; therefore, no one trusts the content. A publisher can infect nodes by sending them a proof-of-time certificate. Each node that gets the certificate sets a Credibility Score to 1 in its local trust table. In this way, the publisher can quickly flood the whole network, which leads to an epidemic in a relatively short time. However, we need trust to increase slowly enough for a malicious publisher to be unable to infect the whole network.

To slow down the spreading process, we can use various mechanisms. In [1], the author proposed a mechanism where each node refuses to get infected unless the previous node signs a certificate. In this way, the publisher has to infect new nodes sequentially, much slower than infecting them in parallel. To control the speed of infection dissemination we can design nodes to sign the certificate after some (e.g., 10-minute) delay. It might be the analog to the time when a host is infected, but is not infecting other people yet. This constraint seems to prevent rapid dissemination, but a smart publisher could request many nodes at a time to sign the certificate and conduct many chains of certification in parallel. To prevent that, we need to force the publisher to start a certification chain from his/her home node assigned by some external mechanism, and each node needs to indicate which node is the next to trust the certificate. In this way, the publisher can not create the certification chain in parallel; it needs to instill infections in a succession. This mechanism is similar to the iterative DNS query; see Fig. 4.1.

There are still some missing points that we need to solve. Although we can control infection dissemination speed, we do not have a mechanism to recover from malicious publications. After a limited period of time the network should eventually notice the malicious publisher and reject the authentication. At the same time, an honest publisher who does not have time constraints can successfully infect the whole network—causing an epidemic and thus authenticating the content.

We therefore need to add a node recovery mechanism. Once the node is infected, it should remain so for a limited period of time. After that time, it recovers, becoming healthy, i.e., distrusts the content. The period of time needs to be carefully designed so the nodes do not recover too fast because it would lead to the nodes getting infected slower than they are recovering.

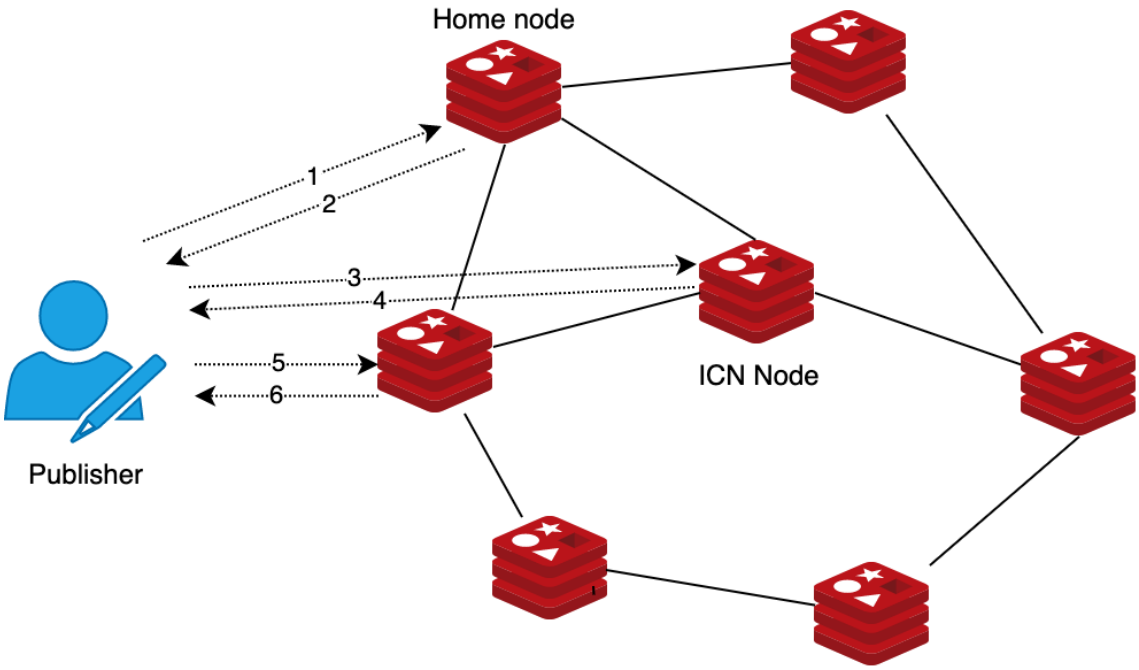


Figure 4.1. Iterative publications of proof-of-time

The requirement that the nodes be infected in succession is inconvenient. It would be much better if, following a limited succession of infections, another mechanism could infect the rest of the network. For that, we can use an additional mechanism for inner infections. Each node gets infected not only by a publisher certificate, but also by the neighbor nodes. In this way, the publisher would be responsible just for starting the initial infection succession, which should embrace the whole network if it is long enough.

5. EPIDEMIOLOGY

To get some background for the graph infection algorithm, we first outline infection processes in the field of epidemiology.

In epidemiology there exist models that represent a spread of disease, such as SI, SIR, SIS, SIRS. Each letter designates the possible state an individual (host or node) can be in, respectively S - susceptible, I - infected, R - recovered. An individual in the S state is healthy, but can contract a disease in contact with someone infected; in the I state is infected and can infect others, and in the R state has recovered. Although this classification is simplified and does not consider inner body mechanisms, it is enough to observe what is happening on a network level. Additionally, the above models entirely ignore contact networks, assuming that each individual has an equal chance to contact anyone else in a unit of time, a so-called *homogeneous mixing* assumption.

5.1. SI Model

SI is the simplest and most primitive model, assuming that there are only two states of a node, susceptible and infected. Let $S(t)$ be the number of nodes in the S state at time t, and $I(t)$ be the number of nodes in the I state. Since the disease-spreading model is a random one, those numbers are not deterministic and can vary among instances of the infection process, even in the same conditions. To get around this problem, we will treat S and I as the average number of susceptible or infected nodes over many instances with identical conditions. In this model, we allow only one kind of state transition, from S to I — the state changes when a susceptible individual meets an infected one. If the total population consists of $n = I + S$ people, then the average probability of meeting a person in a susceptible state is $\frac{S}{n}$. Let β be the chance that individuals contact someone else in a unit of time. Hence an individual has a $\beta * \frac{S}{n}$ chance of contact with a susceptible individual. Since the total number of infected people is I, then the rate of new infections per unit of time is equal to $I * \beta * \frac{S}{n}$.

The SI model can be written as an ordinary differential equation (ODE):

$$\frac{I}{dt} = I\beta\frac{S}{n} \tag{5.1}$$

Accordingly, the decreasing rate of susceptible equals:

$$\frac{S}{dt} = -I\beta\frac{S}{n} \tag{5.2}$$

We can also rewrite the equation to the variables representing fractions of susceptible and infected nodes

$$s = \frac{S}{n}, i = \frac{I}{n} \tag{5.3}$$

Then we can rewrite Eqs. 5.1 and 5.2 as:

$$\begin{aligned} \frac{di}{dt} &= i\beta s \\ \frac{ds}{dt} &= -i\beta s \end{aligned} \tag{5.4}$$

This kind of differential equations are called logistic growth equations, and can be solved with:

$$i(t) = \frac{i_0 * e^{\beta * t}}{(1 - i_0) + i_0 * e^{\beta * t}} \tag{5.5}$$

Wwhere i_0 is the value of i at $t = 0$. In Fig. 5.1 we see the $i(t)$ function where $i_0 = 0.01$, $\beta = 1$.

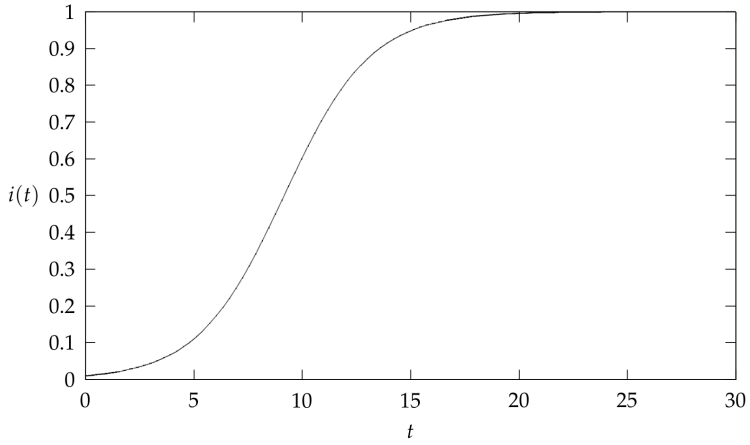


Figure 5.1. Logistic growth function $i(t)$

5.2. SIR Model

SIR model introduces the R state to the SI model. When, in the SI model, an individual gets infected, it remains so forever. SIR model allows infected nodes to recover from the disease after some time. In the real world, this might be because of the immune system fighting the disease. Additionally, once an individual recovers from the infection, it becomes immune to further infections. Therefore in SIR, an individual can change state only from S to I to R. In this mathematical model, we do not distinguish if the R state is obtained by immunization or death, since in both cases, an individual is removed from the potential disease hosts pool. Because of that, the model is also called susceptible-infected-removed.

Call γ the chance of recovering from the I state. Then the ODE system for SIR model becomes:

$$\begin{aligned} \frac{ds}{dt} &= -\beta i s \\ \frac{di}{dt} &= i\beta s - \gamma i \\ \frac{dr}{dt} &= \gamma i \end{aligned} \tag{5.6}$$

To see how this model behaves with some concrete values, consider the COVID-19 epidemic with the following assumptions: three individuals were initially infected at t_0, the total human population is 7.8b people, and no individual was immune to the disease at t_0. Hence,

$$I(0) = 3$$
$$S(0) = 7.8 * 10^9$$
$$R(0) = 0$$

We assume that each infected individual, on average, makes a possibly infecting contact once in two days (assuming homogeneous mixing across the whole globe), and the infection duration is five days:

$$\beta = \frac{1}{2}$$
$$\gamma = \frac{1}{5}$$

The following plot (Fig. 5.2) shows fractions of susceptible, infectious, recovered individuals in the whole population as a function of time.

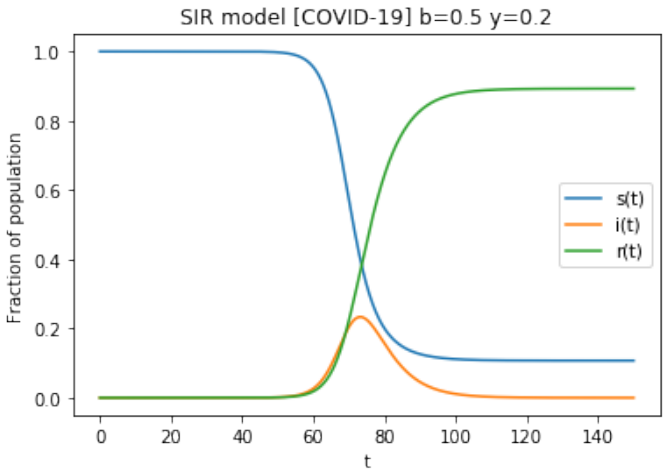


Figure 5.2. COVID-19 in SIR Model for $\beta = 0.5$

If we decrease the number of infections to one in five days, so $\beta = 0.2$, then there is no epidemic (see Fig. 5.3).

The rate of new infections $\frac{di}{dt}$ determines if an epidemic will happen; if it is positive, then there will be an epidemic, and if it is negative, individuals will recover faster than the spread of infection, so there will be no epidemic.

We can calculate the value of $\frac{di}{dt}$ at t_0 and check if the value is negative or positive; the negative value means the decrease of infections from the beginning, so we can be sure that eventually there will be no infections.

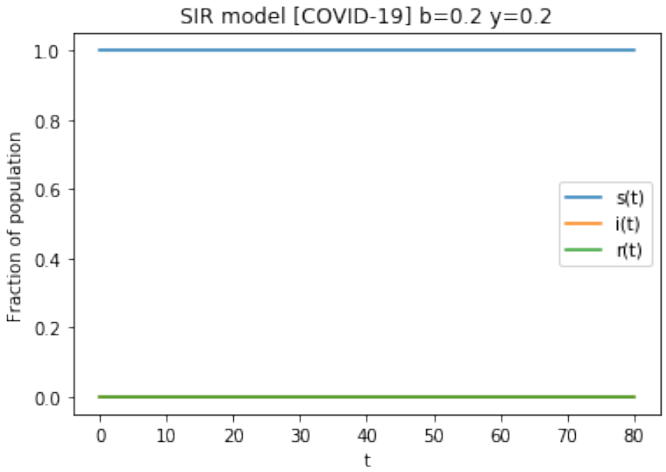


Figure 5.3. COVID-19 in SIR Model for $\beta = 0.2$

The condition determining if an epidemic will happen or not,is thus

$$i\beta s_0 - \gamma i > 0 \tag{5.7}$$

Since the fraction of infections i is never negative, we can omit it to get

$$s_0 > \frac{\gamma}{\beta} \tag{5.8}$$

We can introduce R_0, the Basic Reproduction Number:

$$R_0 = \frac{s_0 \beta}{\gamma} \tag{5.9}$$

which represents the number of individuals that each infected individual can infect before recovering. Thus if $R_0 = 2$ then each infected individual can infect on average two other individual before recovering, so the number of infections grows exponentially, and if $R_0 = 0.5$, then for every two individuals, only one new infection happens, and the number of infections decreases exponentially. $R_0 = 1$ is called the *epidemic threshold*, where the number of new infections equals the number of recoveries, so the total number of infected individuals is always the same and equals the initial number of infected individual I_0.

For example, for seasonal flu the value of R_0 is estimated between 0.9–2.1 [12], whereas for COVID-19 it's estimated to 2.2 in early-stage (January) [13] and recently to 5.7 (April) [14].

To mitigate an epidemic, we can decrease s_0 and β, or increase γ. Since we have little control over the time of recovery (γ), the only way to reduce the impact of infection is to reduce the number of contacts between infected and susceptible individuals (which is why social distancing is so crucial in a fight with epidemics).

5.3. SIS Model

There are some diseases where an individual can get infected many times, e.g., the flu. Therefore there is no Recovered population. This could be because the virus is mutating, or

antibodies do not persist long enough in the organism. For this kind of epidemic, the state flow is $Susceptible \rightarrow Infected \rightarrow$Susceptible, hence the name SIS. For this kind of model, we need to modify the previous equations, so that the fraction γi will go to the S state, instead of the R state.

$$\begin{aligned} \frac{ds}{dt} &= -i\beta s + \gamma i \\ \frac{di}{dt} &= i\beta s - \gamma i \end{aligned} \tag{5.10}$$

We can solve the equations by substituting s in the second equation and solving a Linear Differential Equation:

$$\frac{di}{dt} = i * \beta * (1 - i) - \gamma * i \tag{5.11}$$

After integrating we get:

$$i = (1 - \frac{\gamma}{\beta})\frac{C}{C + e^{-(\beta-\gamma)t}} \tag{5.12}$$

where

$$C = \frac{\beta i_0}{\beta - \gamma - \beta i_0} \tag{5.13}$$

In Fig. 5.4 we can see a plot of the function $i(t)$ for constant $\gamma = 0.2$ and $\beta = 0.5$.

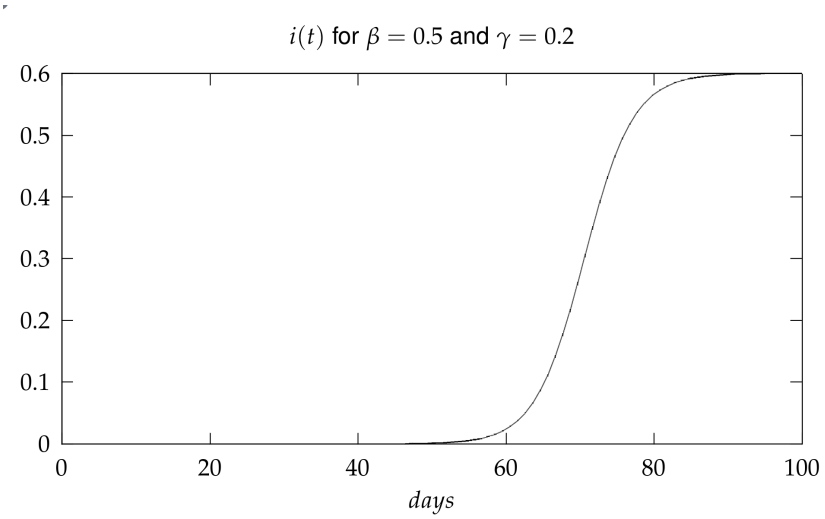


Figure 5.4. $i(t)$ for $\beta = 0.5$ and $\gamma = 0.2$

Also, in Fig. 5.5 we plot the function $i(t, \beta)$ for constant $\gamma = 0.2$ to see how the β influences the results. As we can see, it drives the number of the infected population. If it is less than γ, the epidemic never occurs. Values greater than γ lead to a constant fraction of the infectious population (the rate of catching the infection by the susceptible population is equal to the rate of recoveries in the infected population). The higher the β, the faster is the outbreak, and the higher the infection ratio.

$i(\beta, t)$ for $\gamma = 0.2$

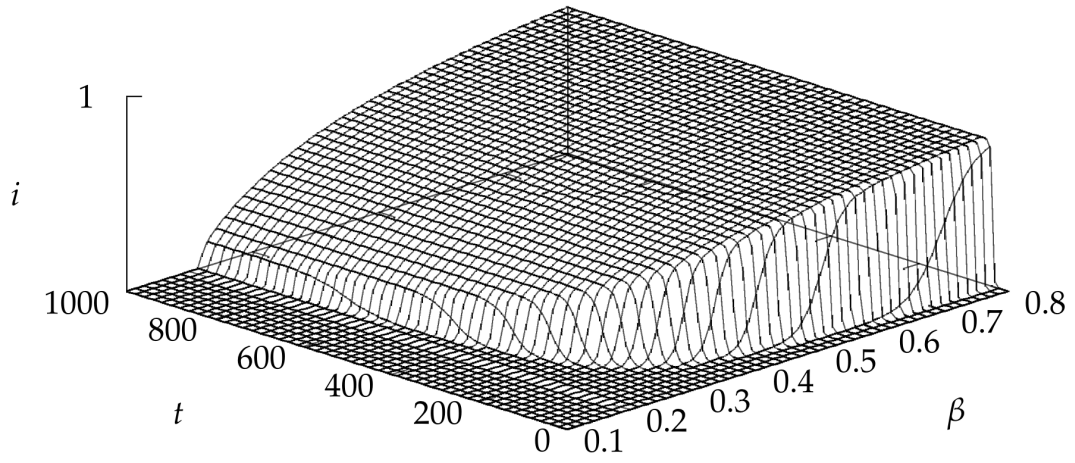


Figure 5.5. $i(\beta, t)$ for $\gamma = 0.2$

5.4. SIRS and SEIRS Models

SIRS model introduces another state change, allowing recovered individuals to lose their immunity. Therefore the state flow is: $Susceptible \rightarrow Infected \rightarrow Recovered \rightarrow Susceptible$. We can write equations for this model by simply adding one intermediate state equation to Eq. 5.10.

$$\begin{aligned} \frac{ds}{dt} &= \delta r - i\beta s \\ \frac{di}{dt} &= i\beta s - \gamma i \\ \frac{dr}{dt} &= \gamma i - \delta r \end{aligned} \tag{5.14}$$

Another, more complex model in terms of the number of states is the SEIRS model. This model introduces an intermediary Exposed (E) state between the S and I states, where an individual is infected but does not spread infection to other people. Such modification is also easy to write by adding another intermediary state to Eq. 5.14.

$$\begin{aligned} \frac{si}{dt} &= \delta r - i\beta s \\ \frac{ei}{dt} &= i\beta s - \gamma e \\ \frac{di}{dt} &= \gamma e - \omega i \\ \frac{dr}{dt} &= \omega i - \delta r \end{aligned} \tag{5.15}$$

Clearly, these models can be easily extended to an infinite number of intermediary states, more accurately reflecting reality. Unfortunately, there is no known analytical solution to such models; they have to be solved by numerical integration of differential equations.

5.5. Summary

The above outlined models hinge on the idealistic full-mixing assumption, where each individual has an equal probability of contacting anyone else. This assumption does not reflect precisely human interaction or distributed computer networks. The β parameter reflects a chance of an individual to contact someone else and spread the infection. In our problem, the network is static, the structure of nodes is fixed; each node has a unique set of neighbors through which the disease can spread. In real life, this might be family members, friends, or coworkers. The chance of contact with the rest of the population, like the chance of meeting two people from two ends of the earth, is minimal. For this reason, we have to consider a different model of infection dissemination. In our model, the infection is an abstract representing trust in content. We will look at how trust graphs are built to understand how to leverage the above mechanisms for our infection spreading model.

6. TRUST GRAPH

When looking into the construction of trust graphs, one can get inspiration from solutions that have worked in human societies for centuries. In his book "Sapiens: A Brief History of Humankind" [15], Yuval Noah Harari, states that the most important feature of human language is rumor; it lets us know which person is or is not trustworthy without having to interact with him/her directly. If our best friend Bob tells us that Charlie is a thief, we do not need to get robbed to be convinced about it. The same applies to an inverse scenario–if Bob tells us that Charlie sells excellent quality products, we are more likely to buy products from him; we are biased towards people about whom we hear positive rumors. Each person we know directly or indirectly gets labeled with some tags. One can be labeled as Helpful, Conscientiousness, or Untrustworthy, while others can be labeled as Unhelpful, Lazy but Trustworthy etc. In this thesis we limit the scope just to the Trustworthiness dimension.

Suppose we have three friends Alice, Bob, and Charlie. Alice and Bob tell us that David is Trustworthy, while Charlie claims that he is not. The decision to label David as Trustworthy or Untrustworthy requires some decision evaluation algorithm. One might for example assume that if there is at least one person who does not trust him, there must be something wrong with him and so label him as Untrustworthy. One can use another, majority-based evaluator: if a majority of my friends trust Charlie, so will I. Another one can slightly generalize this evaluator and say that person is trustworthy, only if a given percentage ξ of my friends trust him.

We now introduce some conventions. A trust relation implies that two nodes trust each other, either by being friends (a friendship relation being given a priori) or have a large enough proportion of mutual friends. Let N be a set of all considered individuals. Let $F_n \subset N$ be the set of all n's friends. Then

$$\%(n_1, n_2) = \frac{|F_{n_1} \cap F_{n_2}|}{|F_{n_1}|} \tag{6.1}$$

is the proportion of n_1's friends who are also n_2's friends. Let's call ξ (where $0 \leq \xi \leq 1$) the minimum value of $\%(n_1, n_2)$ needed to create trust relation from n_1 to n_2.

Coming back to our example, we trust David only if the majority of our friends trust him. We denote trust function as $T : N \times N \to \{true, false\}$, $T(n_1, n_2) = \%(n_1, n_2) > \xi$. To evaluate if we should create trust relation to David, let $\xi = 0.5$ and assume Alice, Bob and Charlie are our friends, and Alice and Bob are David's friends, but Charlie is not.

$$\begin{aligned} T(Me, David) &= \%(Me, David) > \xi \\ &= \frac{|F_{me} \cap F_{David}|}{|F_{me}|} > \xi \\ &= \frac{2}{3} > \frac{1}{2} \end{aligned} \tag{6.2}$$

Then David is **Trustworthy** and we should create trust relation to him.

Intuitively, people with low ξ easily get manipulated and can be called naive; people with high ξ hardly get convinced and can be called stubborn.

Having a trust model, we can describe an algorithm to generate a trust graph.

1. start with initial N_0 nodes and E_0 edges and connect them randomly, such that the graph is connected.

2. each remaining node $n \in N \backslash N_0$, gets connected to one randomly selected node.

3. for each node $n \in N$,

   for each node $f \in F_n$,

   for each node $p \in F_f \backslash F_n$,

   if $T(n, p) = true$, add connection between n and p.

We find some similarities between our graph generator and a trust network formed in PGP systems [16] (with just one level of indirect trust); therefore, we call this algorithm the Web of Trust generator.

In Fig. 6.1 we see the resulting trust graph for $N = 500$, $N_0 = 10$, $E_0 = 20$ and $\xi = 0.5$, and its corresponding histogram of node degree in Fig. 6.2

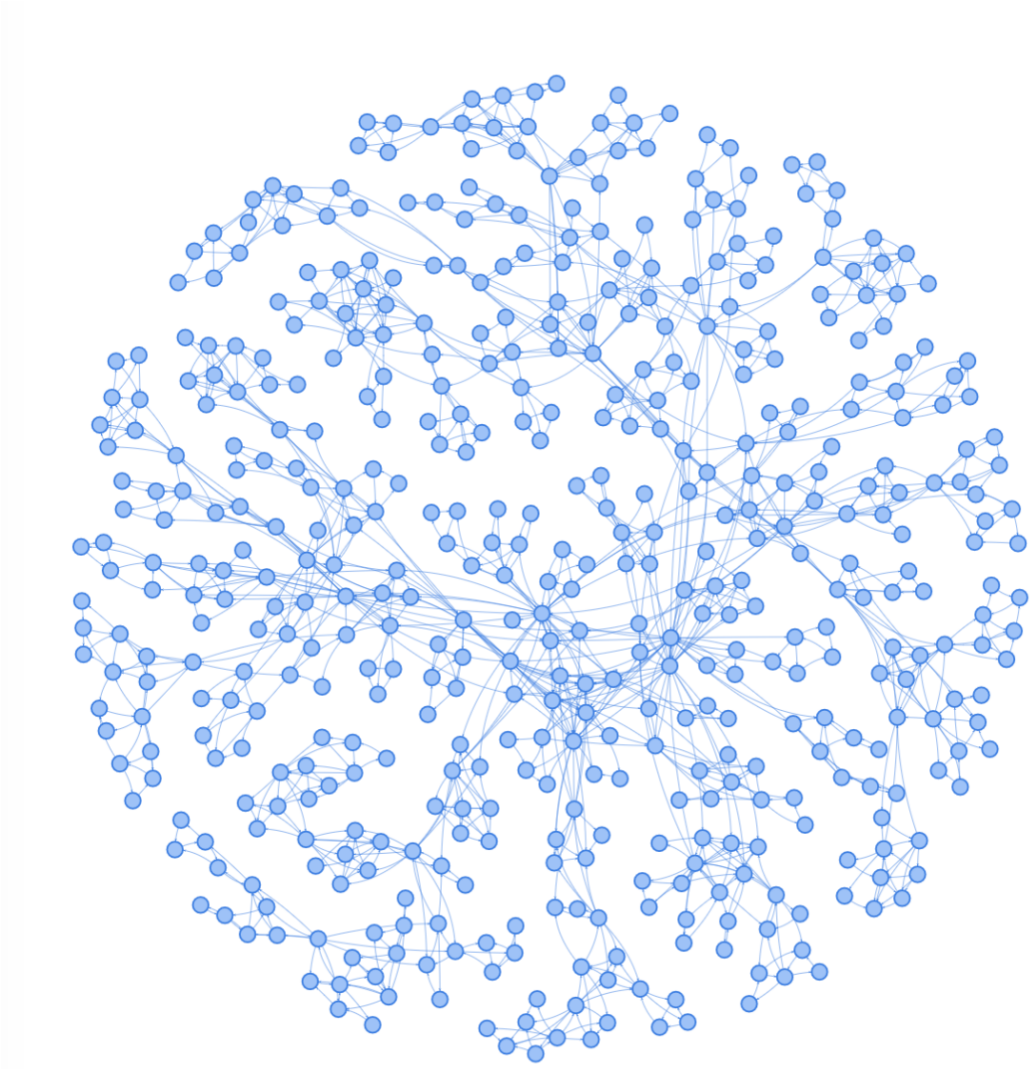


Figure 6.1. Trust Graph generated by Web Of Trust algorithm for $N = 500$, $N_0 = 10$, $E_0 = 20$ and $\xi = 0.5$

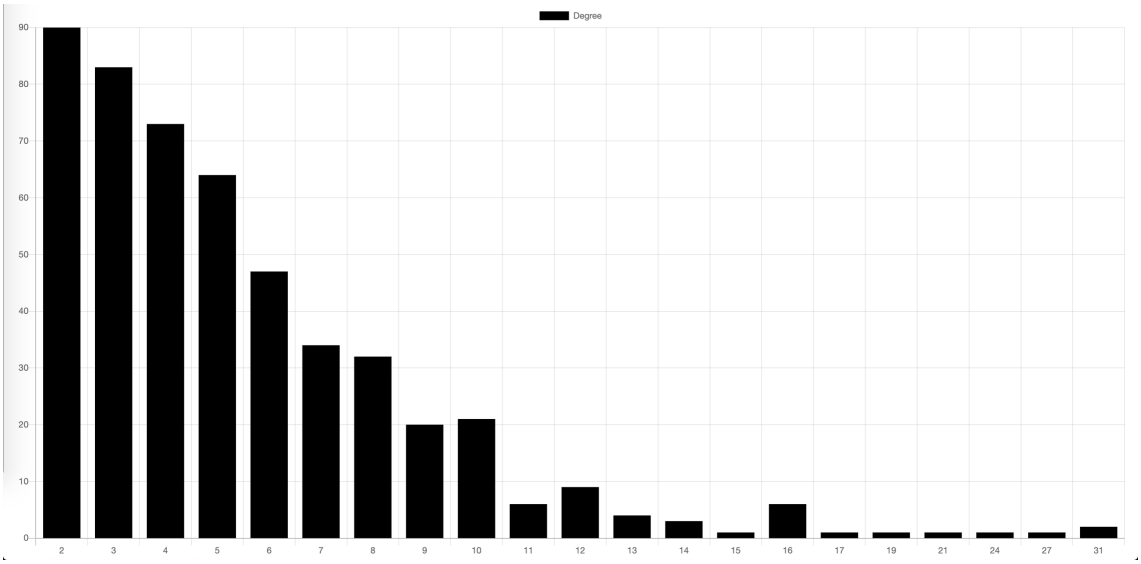


Figure 6.2. Degree histogram generated by Web Of Trust algorithm for $N = 500$, $N_0 = 10$, $E_0 = 20$ and $\xi = 0.5$

6.1. Other trust graph generators

Probabilistic duplication Another trust graph generator used in [1] is based on probabilistic duplication and works as follows:

1. start with initial N_0 nodes and E_0 edges and connect them randomly, such that the graph is connected
2. add new node n_1 and connect it to one randomly selected node n_2 in the current graph.
3. connect n_1 to n_2's friends, to each with probability ϕ. Repeat steps 2 and 3 $|N \backslash N_0|$ times.

The most important advantage of this generator is the fact that it produces almost scale-free graphs. Figure 6.3 shows the graph generated by this method and Figure 6.4 shows the histogram of node degree.

Random generator A random graph generator is the simplest one. Take N nodes, E edges and connect them randomly, such that the graph is connected. Although random trust graphs are far from real-world trust graphs, we use it for comparison. Figure 6.5 shows the graph generated by the method and Figure 6.6 shows the histogram of node degree.

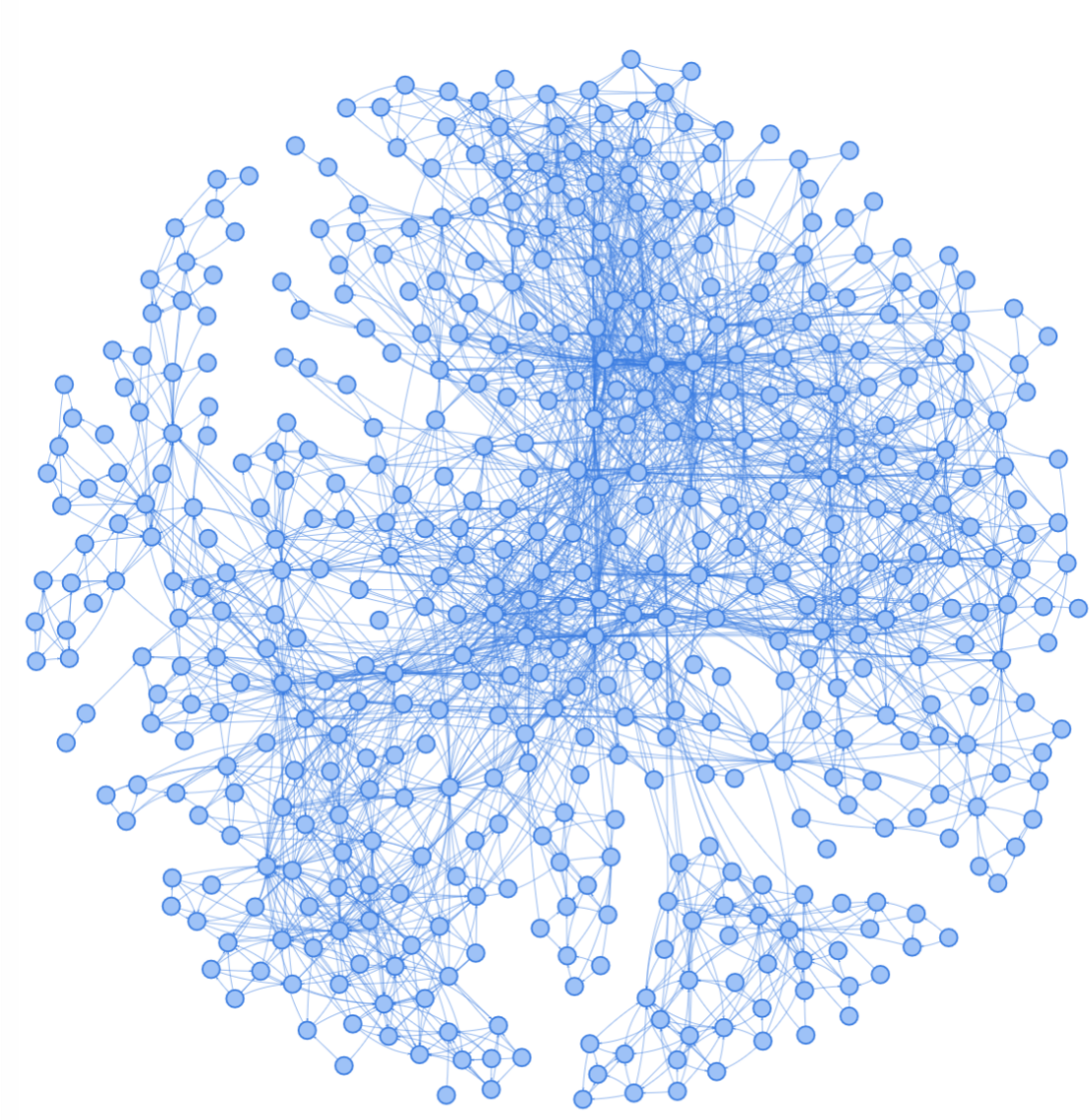


Figure 6.3. Trust Graph generated by Probabilistic Duplication algorithm for $N = 500$, $N_0 = 10$, $E_0 = 20$ and $\phi = 0.5$

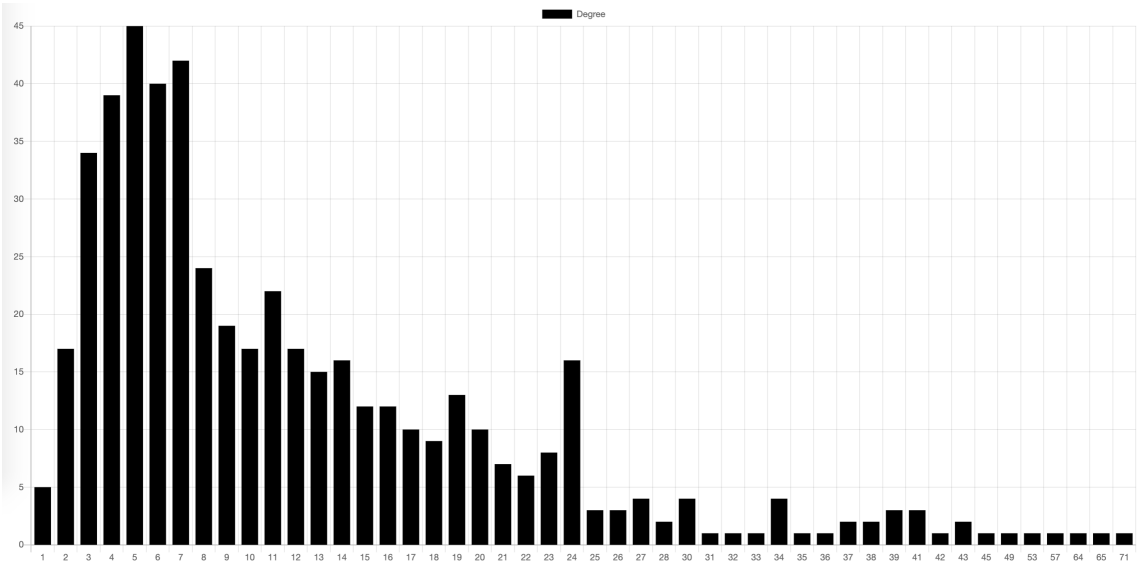


Figure 6.4. Degree histogram generated by Probabilistic Duplication algorithm for $N = 500$, $N_0 = 10$, $E_0 = 20$ and $\phi = 0.5$

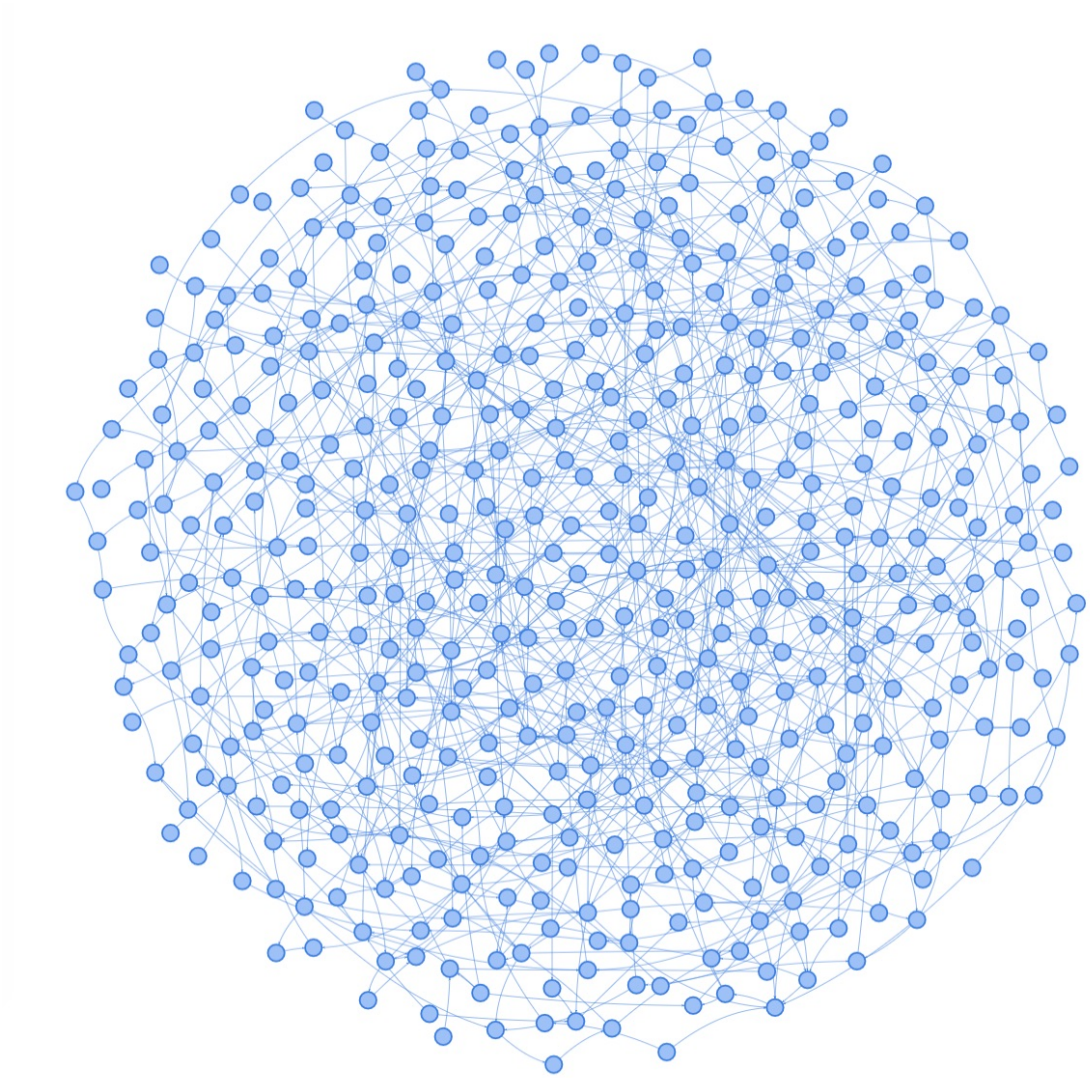


Figure 6.5. Random graph generated by Random generator for $N = 500$ $E = 1000$

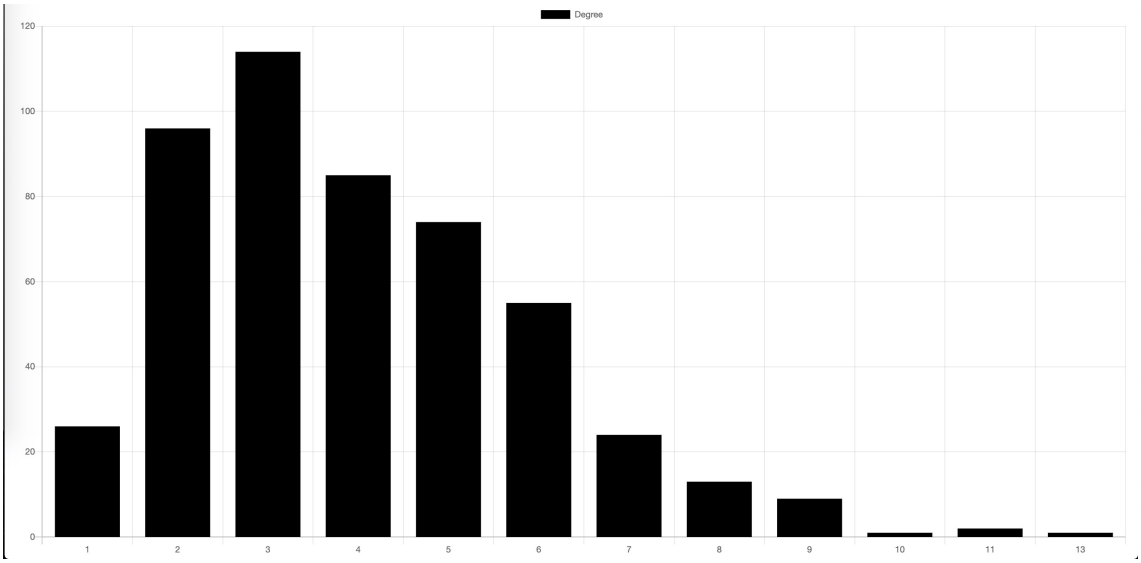


Figure 6.6. Degree histogram generated by Random generator for $N = 500$ and $E = 500$

7. STOLEN CREDENTIALS PROBLEM

ICN authentication model is based on credentials. A unit with access to credentials can publish authenticated data, and the rest of the network has no way to verify the content authentication other than checking signature validity (created with credentials). Credentials can be loosely categorized into: cold/offline and hot/online. Cold/offline credentials, e.g., private keys, have very long (days or months) or indefinite expiration time; they are typically stored on hard drives and rarely leave the device. Hot/online credentials, e.g., access tokens, are created for a limited period (minutes or hours) and are often transmitted over a communication network. Stolen credentials is a serious problem that multi-factor authentication methods try to mitigate, but in this thesis we assume that even such methods cannot solve. In conjunction with ICN caching, the problem can lead to destructive consequences, because ICN protocols do not provide data revoking/removal functions. A malicious publisher that gains access to stolen credentials can publish authenticated data that can stay in a node's cache for a long time.

For cold/offline data breaches, we can find reports (DBIR - Data Breach Investigations Report 2020 [17]) showing that 45% of the attacks are caused by hacking, 22% by misconfiguration, 22% by phishing. 17% caused by malicious software, 4% by misuse by authorized users, and 4% by physical interaction.

Most of the attacks (70%) are performed by external entities motivated financially (86%). Also, most attacks are targeted at big corporations (72%). In most of them (58%), the data breach includes users' personal information.

From the historical data in Fig. 7.1 it follows that the most visible drop occurs for Trojan

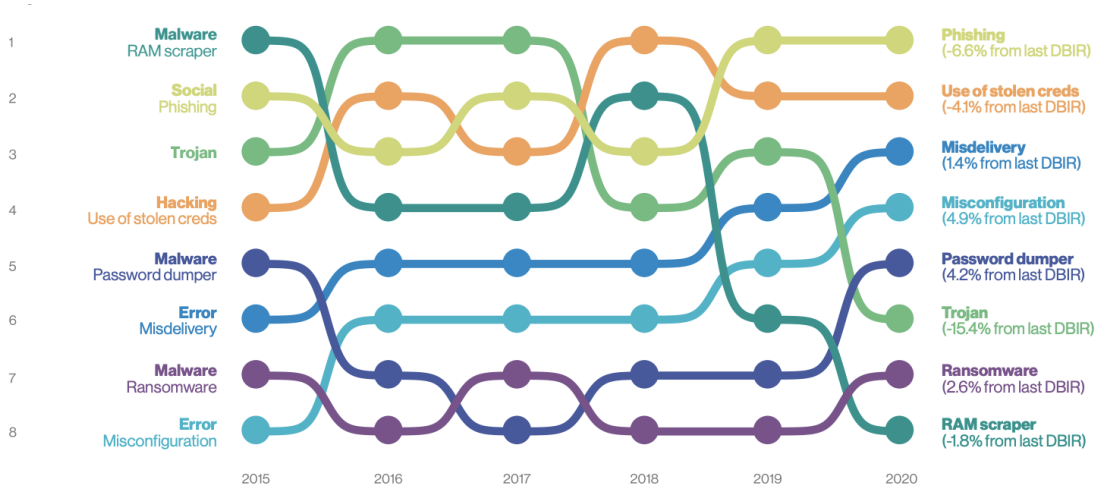


Figure 7.1. Change of breaches over time. Source: Data Breach Investigations Report 2020 [17]

horses—from 50% in 2016 to 5.6% in 2020. A similar drop occurs for RAM Scrapers, which search the operating system memory for potential confidential data. On the contrary, the highest rise can be noted for misconfiguration and accidental data leakage. Yet the highest popularity is still observed for phishing attacks and usage of stolen credentials. In Fig. 7.2, one sees that for 60% of the incidents, *the breach was discovered in less than one day*, and this trend is increasing—more incidents are discovered in less than one day. 25% of incidents were discovered in one or more

months. However, it must be noted that some breaches might not have been discovered yet, so the value may be underestimated. In this same report we can find that the number of discoveries has increased due to Managed Security Service Providers (MSSP), which are obligated to publish such breach incidents.

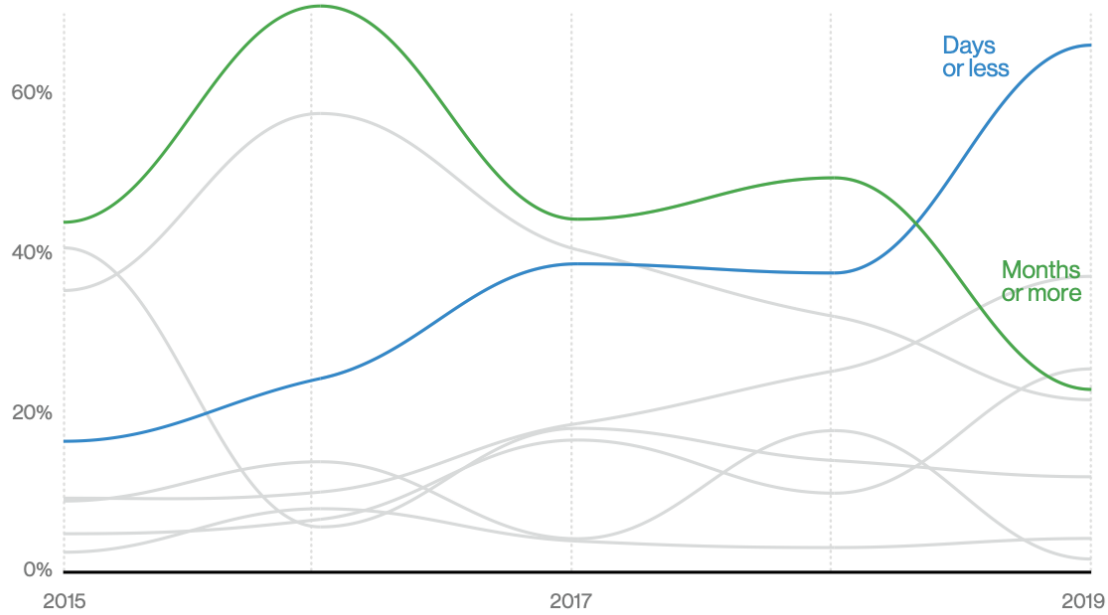


Figure 7.2. Discovery over time in data breaches. Source: Data Breach Investigations Report 2020 [17]

Credentials such as access tokens (e.g., JWT tokens) used in communication between two parties have some lifetime duration, after which they expire. In the IETF RFC 6819 document [18], we can find that the suggested lifetime for such credentials ranges from minutes to hours depending on the risk associated with token leakage.

8. GRAPH INFECTION

The Graph Infection (GI) algorithm proposed in [1] is similar to the SEIRS model, but some changes were made. There are two healthy states–Susceptible and Quarantine, and two Infected states–Acute and Recoverable. The state transition machine is presented in Fig. 8.1.

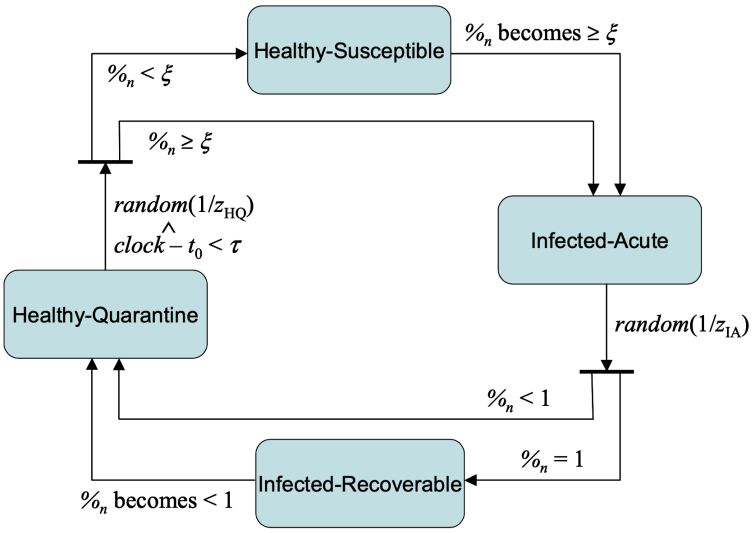


Figure 8.1. Finite state machine at a node. Source: Mitigating Time-Constrained Stolen-Credentials Content Poisoning in an NDN Setting [1]

The four states can be described as follows:

- Susceptible nodes are healthy—they do not propagate infection, but can get infected, whereupon change state to Acute.
- Acute nodes are infected—they inflict infection propagation and switch state to Recoverable or Quarantine depending on how many of their neighbors are infected.
- Recoverable nodes are infected but can switch to Quarantine if some of their neighbors become healthy.
- Quarantine nodes are healthy and can switch state to Susceptible or Acute, depending on how many of their neighbors are infected.

Nodes can get infected by either:

1. external infection – a content publisher can always infect his/her home node, and then successive nodes at fixed time intervals.

2. internal infection – when the fraction of infected neighbors surpasses the ξ threshold.

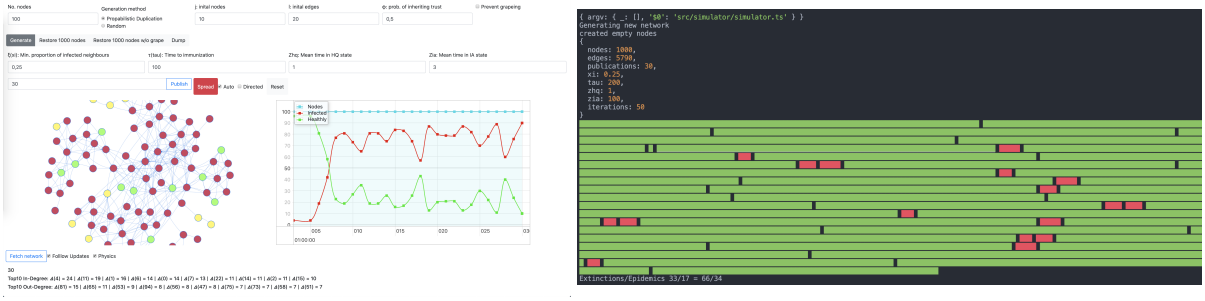
Initially, all nodes are healthy (distrust new content). When a publisher wants to authenticate new content, he/she starts the external infection process by infecting his/her assigned home node. The home node imposes a fixed delay, after which it issues a new certificate allowing the publisher to infect the next node selected at random. The publisher infects the next node by presenting the certificate signed by the previous node. This process is repeated until an epidemic occurs or the intruder loses access to the stolen credentials. To give this process some momen-

tum, we allow the infected nodes to infect others in a process of internal infections. In this way, the publisher does not need to coordinate the whole process until the network reaches epidemic, but just starts a ”chain reaction”, and if he/she does so with sufficient power, an epidemic will occur.

Internal infections work as follows: a node n in the Susceptible state switches to the Acute state by being infected by a sufficient proportion of its neighbors, specifically if the fraction of its infected neighbors $\%_n = \frac{|I_n|}{|N_n|}$ reaches the threshold ξ, where I_n is the set of infected n's neighbors and N_n is the set of all n's neighbors. For simplicity let us measure time in units called *cycles*, a cycle being long enough for a node to infect a neighbor node. To give the process some momentum, the Acute and Quarantine states are introduced. A node has to spend some random time before leaving either state, denoted $random(\frac{1}{Z_{IA}})$ and $random(\frac{1}{Z_{HQ}})$, respectively, where Z_{IA} is the mean number of cycles in the Acute state, and Z_{HQ} is the mean number of cycles in the Quarantine state. Depending on $\%_n$, a node leaving the Acute state switches either to Recoverable—if $\%_n = 1$, or directly to Quarantine—if $\%_n < 1$. In the Recoverable state, the node is still infected, but as soon as one of its neighbors recovers, it switches to the Quarantine state. The Quarantine state is similar to Acute except that a node can be locked in this state forever if it gets immunized. The immunization is acquired after τ cycles from the first infection on the node, where τ is a fixed parameter working as a timeout, preventing an endemic (partial epidemic) of the network. If an endemic were to prevail for a long time, it would mean that the publisher's proof-of-time was too weak to reach an epidemic, but too strong for all the nodes to recover.

As we discussed earlier, there is no known analytical solution to such a graph model; therefore, it needs to be computer simulated.

8.1. Simulator



(a) Visual simulator (b) Fast simulator

Figure 8.2. Simulators preview

We have developed two simulators, the first for visualization and the second for fast calculations. The visualization simulator helps to better understand the processes on graphs and

find potential problems. The fast simulator allows to perform various calculations on different input data in a reasonable time. The source code for both simulators is available at https://github.com/stasbar/ProofOfTime-Auth. Both simulators are presented in Fig. 8.2. The visual simulator is accessible at masti.stasbar.com. Both simulators allow us to generate the trust graph using three different generators: random, web of trust, and probabilistic duplication discussed in Chapter 6 with customized parameters: number of nodes, number of nodes and edges in the initial kernel, and ϕ, used in the web of trust and probabilistic duplication to customize the density of the connections. After the graph is generated, one can save and restore it to always work on the same graph during further research. When the graph is loaded, we can configure the parameters ξ, τ, Z_{HQ}, Z_{IA} and the number of external infections (i.e., publications or proofs-of-time). The fast simulator outputs the results of simulations; the green color indicates that the network ended up in an extinction (all nodes have recovered) and the red color indicates an epidemic; the length of the block indicates the number of cycles needed to reach one of these outcomes. The visual simulator provides a live preview of the graph and a chart of the ongoing infection process.

8.2. Observations

We run the simulator against many different configurations to find the influence of each parameter. The algorithm's probabilistic nature forces us to run the simulation multiple (in our case, 200) times for each configuration to get credible results. We run the simulation on 1000-node graphs generated using both web of trust (WoT), random(RND), and probabilistic duplication (PD) generators, but for each simulation, we use the same generated graph. First we investigate the Z_{IA} parameter by assuming all remaining parameters constant: $\xi = 0.25, Z_{HQ} = 1, \tau = 200$ with the maximum number of external infections (publications) equal to 300. Figure 8.3 shows

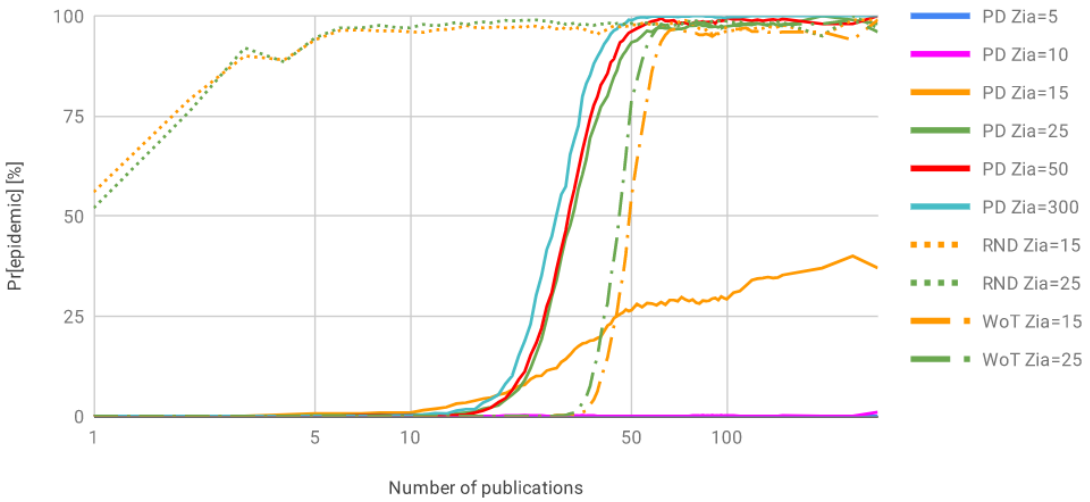


Figure 8.3. Influence of Z_{IA} parameter

how Z_{IA} influences the probability of reaching an epidemic, (Pr[epidemic]), against the number of external publications (proof-of-time). It does not come as a surprise that the higher Z_{IA}, the longer nodes stay infected, and the faster an epidemic is reached. The PD and WoT graphs

produce similar shapes. The WoT graph starts to reach an epidemic a little later. The RND graph is completely different; even with a low Z_{IA}, it reaches an epidemic with just one publication. It happens because most of the nodes have only one or two edges. As a result, one infected node can infect most of its neighbors. In contrast, the PD and WoT graphs often produce nodes of a high degree that are hardly infected by their neighbors. For all graphs with Z_{IA} starting from 25, the chance of reaching an epidemic in near 100% starts at about 50-80 external publications. E.g., if we assume the required proof-of-time amounts to 12h, the publisher needs to publish the proof-of-time each 9 to 14 minutes until an epidemic becomes certain.

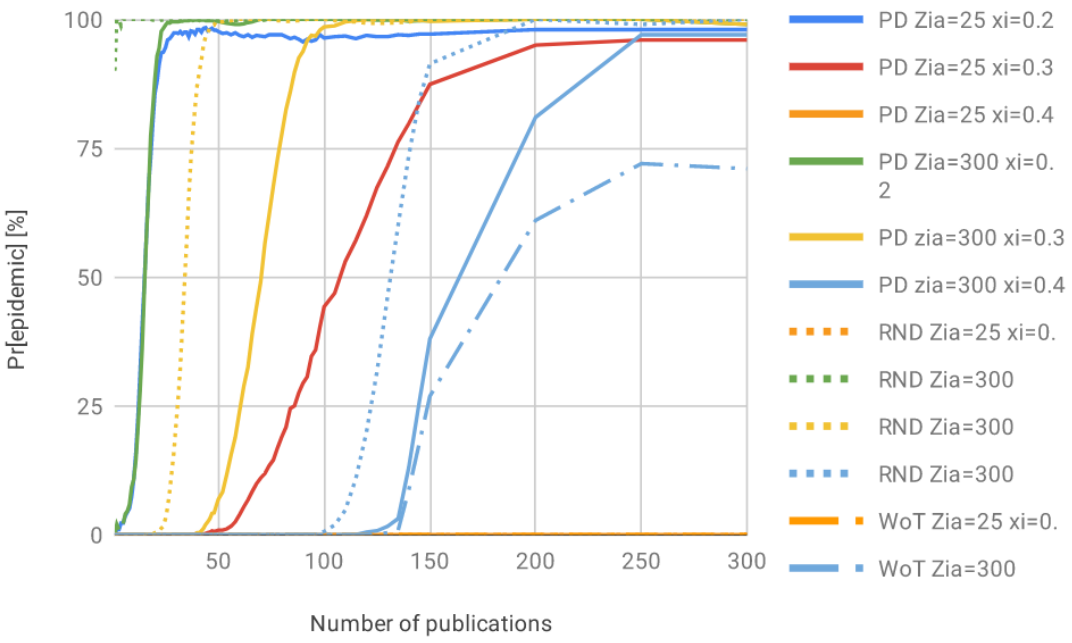


Figure 8.4. Influence of ξ parameter

In a similar way, we test the ξ value. Figure 8.4 shows the Pr[epidemics] against the proof-of-time using different ξ values. We can easily see that ξ is the most sensitive parameter. For values starting from $\xi = 0.5$ not a single epidemic simulation occured for any graph generator.

We notice that the parameters Z_{ia} and xi can completely change the outcome depending on the graph type. Their universal values may be hard to find if we want a solution that works on all types of trust networks. Moreover, if we want to set a fixed number of external publications needed to achieve an epidemic, those values should be dynamically changed according to the size of the network. For example for $\xi = 0.25$, $Z_{IA} = 25$, $Z_{HQ} = 1$, $\tau = 200$ the probability of epidemic is significantly influenced by the size of the graph (see Figure 8.5). Therefore we believe that those parameters should be dynamically adjusted with the network structure changes.

An interesting observation relates to the average number of cycles the simulation needs to finish at some final outcome given Z_{IA} (see Figure 8.6). The average number of cycles needed to reach an extinction grows rapidly until the probability of an extinction is greater than the probability of an epidemic. (Intuitively, the equilibrium point at which the network has the same chance to reach an extinction and an epidemic corresponds to the longest a simulation run can get.) A grow-

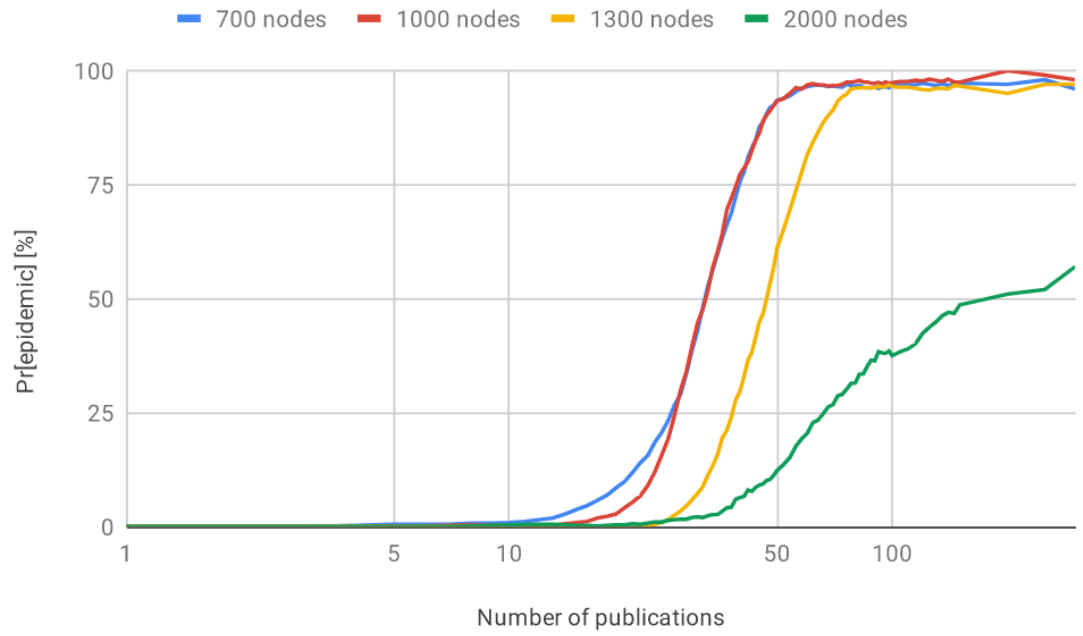


Figure 8.5. Influence of network size

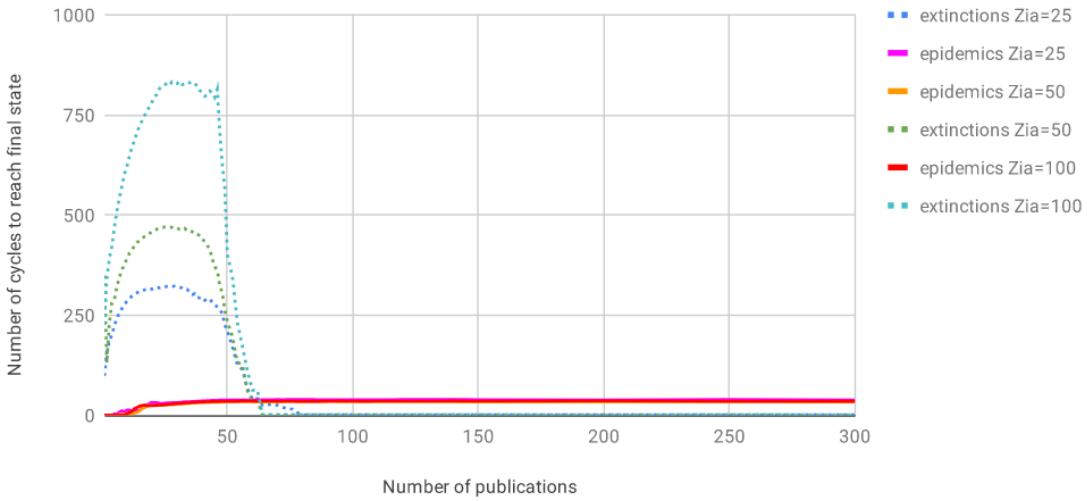
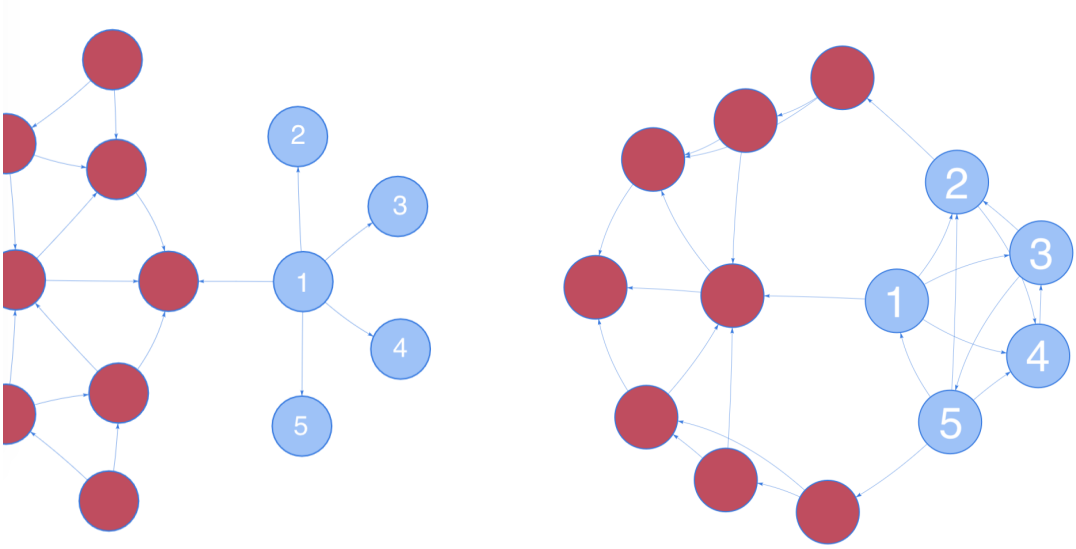


Figure 8.6. Average number of cycles to reach a final outcome

ing Z_{IA} first increases this number by keeping the nodes in the infected state for a higher number of cycles, then decreases it to zero—indicating that the network can not reach an extinction anymore, or when it does, it reaches it quickly. The average number of cycles to reach an epidemic is stable at 33 to 38 cycles, perfectly matching the average number of publications needed to reach an epidemic. Intuition tells us that if the network has not reached an epidemic by the time of the last publication (or few cycles after it), there is no chance of reaching an epidemic–the number of external infections was too small.

Another thing that looks strange is that even with many publications, there still exist simulations where the network does not reach an epidemic. To find an explanation of this riddle, we run visual simulation hoping for a clue. This provides us some interesting feedback–some ”unfortunate” graph instances and ξ values prevent the whole network from reaching an epidemic

even with a large number of external infections (proofs-of-time). If in a trust graph there exists a *defensive alliance* $A \subseteq N$ of connected nodes where each node $v \in A$ is connected to fewer than $\xi * |N(v)|$ nodes outside A, then such an alliance prevents than epidemic. If we assume $\xi = \frac{1}{4}$ and the graph structure visible in 8.7a, then the infected nodes (in red) are not able to infect the healthy



(a) (b)

Figure 8.7. Defensive Alliance

ones (in blue), especially v_1, which is the single node connected to the outside of A. It happens because the healthy node v_1 has five neighbor nodes, of which four are healthy ($\{v_2, v_3, v_4, v_5\}$), hence the one infected neighbor node is not enough to reach the $\xi = \frac{1}{4}$ threshold at v_1. Let us take a different graph where defensive alliance A is connected to rest of the graph $N \ A$ through more than one node as shown in 8.7b. Nodes from A cannot be infected since there are not enough connections from outside A to reach the ξ threshold. A general defensive alliance is defined by:

$$\forall v \in A, \frac{|N(v) \cap A|}{|N(v)|} > 1 - \xi. \tag{8.1}$$

As a result, the rest of the network stays in the Acute state until the nodes connected to A change their state to Quarantine (notice that their $\%_n$ are less than 1), which then propagates throughout the rest of the network leading to an extinction.

To eliminate defensive alliances, we could modify the trust graph generator, but that would violate our assumption that the structure of the trust is taken as a given. Lowering ξ might be another option, but in this way, we would speed up the infection process on the whole network, which is an unwanted effect.

The problem of finding a defensive alliance is NP-complete [19]. There is no known solution to finding defensive alliances or even answering in polynomial time whether there exists one in the graph. Therefore the only way of locating defensive alliances in the graph is an exhaustive search. The problem gets even worse if we assume that the network is open, i.e., at any moment nodes can join or leave.

9. CONSENSUS

If we look at our problem from a different perspective, we notice that we are trying to achieve a consensus among all the nodes about whether the given piece of content is authentic or not. In the graph infection (GI) algorithm, the consensus is dictated by the number of external infections. Too low a number leads to an extinction, whereas a high enough leads to an epidemic. Both an epidemic and an extinction can be considered consensus decisions. In an epidemic, the whole network generates a positive decision about the content authentication, while in an extinction, the network decides the opposite.

If we evaluate GI in terms of the consensus problem, it turns out that it struggles with faulty nodes. In GI, each node controls the time after the next external infection is allowed. If a node fails, it can allow immediate propagation or not allow it at all. Additionally, the consensus is not deterministic—there is no guarantee that the network will reach the same final occurrence for each simulation run with the same initial conditions. Although the consensus is achieved using timeout (the τ parameter specifying the time to immunization), the time to reach a consensus is non-deterministic.

If we consider *Byzantine failures*[20] that allow nodes to act arbitrarily (e.g., send corrupted, malicious, or conflicting information to other nodes) then a node can be stuck in either a healthy or infected state, ignoring the state of its neighbors, hence a single faulty node can change the decision of all non-faulty nodes. In consequence, it can prevent the whole network from reaching an extinction or an epidemic. Consider again the graph in Fig. 8.7a where all nodes are healthy, node 1 is an adversary and falsifies its state as infected; in this way, it infects nodes $2, 3, 4, 5$ which are entirely dependent on node 1; therefore, they will stay infected as long as node 1 decides.

We evaluate the presented GI protocol using three known consensus requirements: Validity, Agreement, and Termination.

1. Validity: any value decided upon must be proposed by one of the nodes.

2. Agreement: all non-faulty nodes must agree on the same value

3. Termination: all non-faulty nodes eventually decide.

Validity and Agreement specify what *must not* happen—there cannot be a situation where the consensus is reached on a value that was not proposed by any node (in our case, neither an epidemic nor an extinction). Also, there must be no non-faulty nodes that decide other than the rest of the nodes. In the distributed systems literature [21] those properties are called *safety*. On the other hand, Termination specifies what *must* happen—at some point, all non-faulty nodes must reach a consensus. In the literature, this property is also called *liveness*.

The Validity requirement is stated to avoid trivial consensus algorithms like ”always choose 0,” which is not the case in the GI protocol.

The Agreement requirement is satisfied owing to the timeout parameter τ that covers some corner cases involving defensive alliances.

Termination is also satisfied—if the network cannot reach a consensus, then after τ number of cycles, nodes get immunized leading the whole network to the final occurrence. The longer the simulation takes, the more likely the network is to reach an extinction and less likely to reach an epidemic.

We have seen that in the case of Byzantine failures, neither fault-tolerance nor safety is satisfied. One single faulty node can prevent non-faulty nodes from deciding on the same value, breaking the safety requirement using one single faulty node. It is probably not impossible to extend the GI algorithm such that it satisfies those properties. If the nodes could gain more awareness of the network state, they could detect faulty nodes and therefore become fault-tolerant. However, it is worth looking at alterntive solutions.

There are a variety of consensus algorithms that differ in the allowed types of node failure (Byzantine or not), synchrony (Asynchronous or Synchronous), the number of tolerated failure nodes, authentication (whether messages have to be signed by their authors), and the number of cycles to achieve consensus.

Instead of looking at the original Stolen Credentials problem as a consensus protocol design problem, we propose a solution that can be deployed on top of an already existing consensus protocol. We search for protocols resilient to Byzantine failures because, for an Internet-level protocol managed by many organizations, we cannot assume that all nodes will act reliably and honestly. Additionally, the protocol must allow open membership–nodes should be able to join and leave the network at any time.

10. BLOCKCHAIN

Consensus protocols that work with Byzantine-failure nodes and allow open membership are used in open blockchains. By an open blockchain we understand one where nodes can freely join the network and participate in the consensus protocol. Examples of such blockchains are Bitcoin [22], Ethereum [23], and Stellar [2]. Bitcoin uses a proof-of-work consensus algorithm where the computational power dictates the contribution to the consensus decision. Ethereum plans to switch to a proof-of-stake consensus where the amount of cryptocurrency dictates the contribution to the consensus decision. Stellar uses Federated Byzantine Agreement where the trust dictates the contribution to the consensus decision. We decide to use the last one, and the rationale for doing so is described in Sec. 10.8.

Data stored in a blockchain are immutable—once written, they cannot be changed. Data are also secure against the intruders (as long as they do not control most of the consensus means). Data types differ in different blockchains, most of which store transactions to update a global ledger. Some blockchains, like Ethereum, also allow storing smart contracts[1], though we do not see any benefit of using them in our system. We use a simple transaction model with a slight modification that prevents pre-signing of proof-of-time transactions. A full description is provided in Sec. 10.6.

10.1. System proposition

We propose a system where all or part of the nodes in the ICN network participate in a blockchain consensus protocol—securely storing a common database consisting of proof-of-time claims, along with the Content Store. Each node can be sure that the rest of the network stores the same version of the blockchain database. But the most vital feature of our system is that by using a blockchain, we ensure that publishers can publish only one proof-of-time at a time. Therefore, to authenticate the content, they must prove their access to credentials over a long enough period of time, unaffordable for a malicious publisher but affordable for an honest one. This is the statement we base our authentication mechanism on. A publisher wishing to confirm content authenticity must submit a carefully prepared transaction to the blockchain network. The transaction—a proof-of-time claim—must include:

- A hash of the content to authenticate.
- A signature on the claim that proves access to appropriate credentials.
- The publisher's public key used to verify the signature and identify the publisher.
- A hash of the previous block to prevent pre-signing of transactions.

A graphical illustration of such a transaction is presented in Fig. 10.1.

The publisher submits the transaction to the nearest node, which includes it in a block and broadcasts to the rest of the network. Once all network nodes approve the block (through the consensus protocol described in Sec. 10.8), the publisher can create the next proof-of-time transaction pointing to the previous block's hash. The process is repeated until the required Credibility

[1] Smart contracts are scripts that are executed on virtual machines on all nodes and use blockchain as a persistent storage

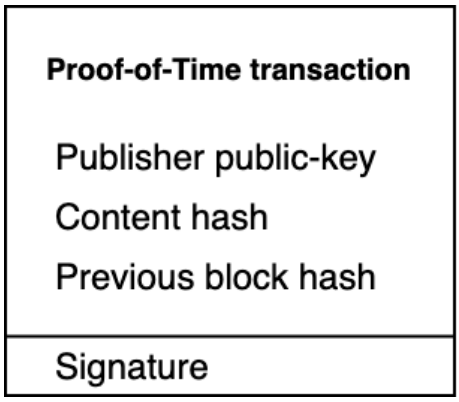


Figure 10.1. Structure of proof-of-time claim

Score (described in Sec. 10.3) is reached. In the sequel, we describe the details of the system based on blockchain.

10.2. Blockchain layer

In our proposal, network nodes play two roles: of an ICN node participating in routing and content caching, and of a blockchain node participating in the consensus protocol and blockchain storage (see Fig. 10.2). Not every node has to play these two roles–the ones not connected to end-users might not participate in the blockchain, since they do not claim content trustworthiness. The blockchain layer could also be managed by entities other than network nodes, thus separating the content distribution (transport) layer from the trust layer, cf. Fig. 10.3. The ICN nodes could be relieved from the trust network overhead, which could make them simpler. Also, the trust system would be more portable; it could be used in different ICN solutions, and even in legacy systems, since the developed trust does not depend on the ICN architecture but on the hash of the content and publisher credentials. The blockchain trust network could be hosted by more powerful devices and possibly different organizations—achieving separation of concerns, which is always desirable in the long term. For the rest of the thesis, we assume the separation of roles (ICN and blockchain), but the decision if one node should play one or two roles is up to the implementation.

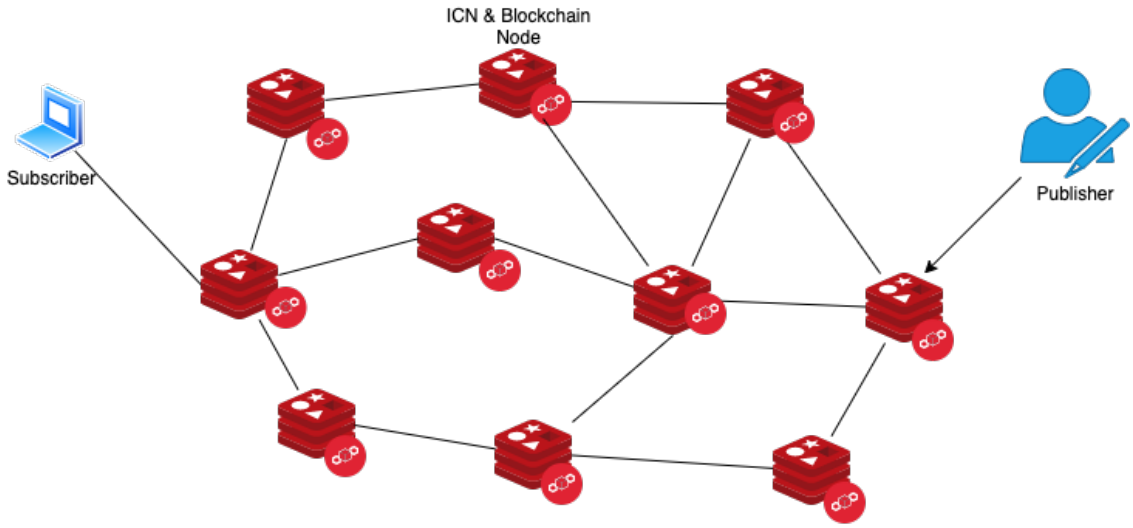


Figure 10.2. Combined layers

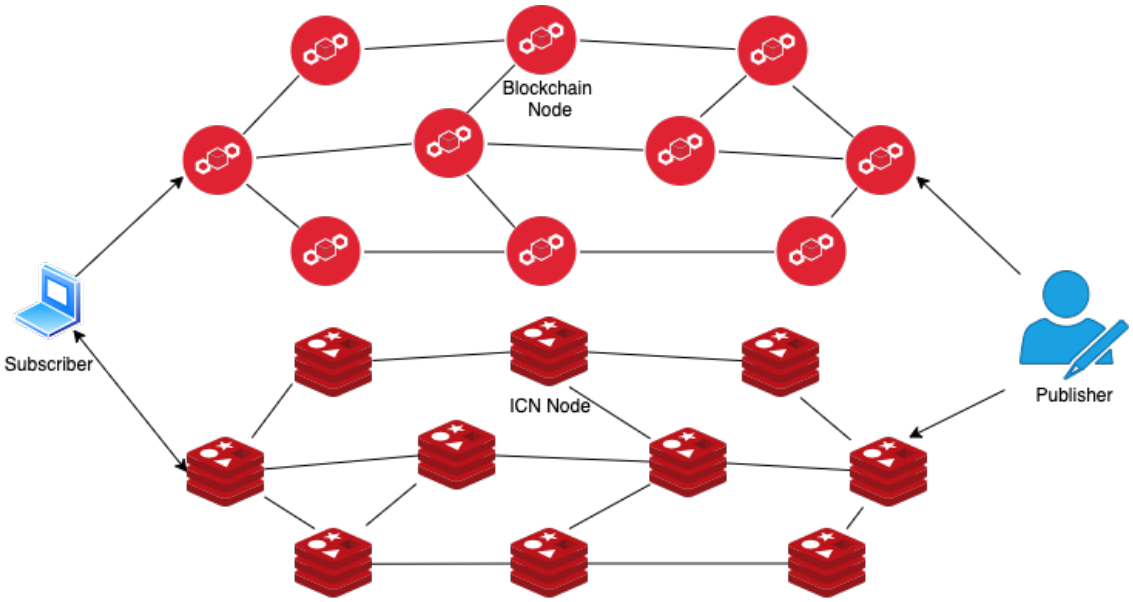


Figure 10.3. Separated layers

10.3. Credibility Score

In the GI protocol, there are two states of content authenticity: authenticated and not-authenticated. We believe that this is limiting. Content like, e.g., weather forecasting, should not be authenticated for the same amount of time as data in a banking website. Therefore we propose a more flexible model, where authentication can be acquired progressively via Credibility Score, mentioned in Section 3, which increases authentication granularity. Different thresholds should be used for different content types. For example, if three trust thresholds are used: low, medium, and high; then, we can require 10 minutes, 2 hours, and 12 hours of proof-of-time respectively. Each time the content is published to the network with stolen credentials, their owner can be notified about it, and has a certain time to revoke the credentials and halt the malicious authentication process.

10.4. Publisher flow

A publisher wishing to publish content on the ICN network and then authenticate it over the blockchain network needs to take the following steps:

1. Create a Named Data Object (NDO), which requires signing the content using the publisher's key pair.
2. Compute a hash of the NDO.
3. Create a proof-of-time transaction, which includes the hash of the NDO, hash of the previous blockchain block, and the signature proving access to the private key.
4. Publish the proof-of-time transaction to the blockchain node.
5. Repeat steps 3 and 4 until the Credibility Score is reached.

The complete flow of publishing the content to the ICN node and blockchain node is presented in Fig. 10.4

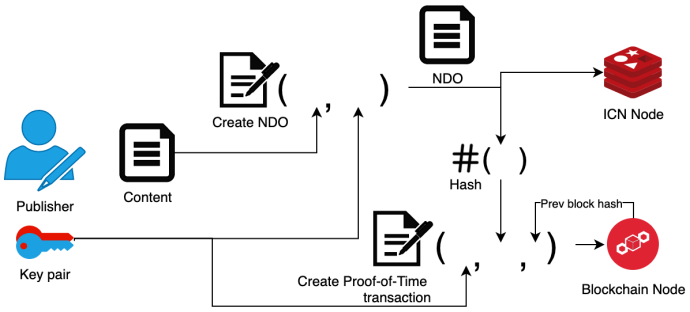


Figure 10.4. Flow of publishing content to ICN and proof-of-time to blockchain

10.5. Subscriber flow

A subscriber willing to fetch the content from the ICN network and verify the authenticity from the blockchain has to take the following steps:

1. Send an Interest packet with a content name to the nearest ICN node.
    - subsequently, the ICN node returns a Data packet with content, signature, and the publisher's public key.
2. Send a Credibility Score request along with the hash of the content and the publisher's public key to the blockchain node.
    - subsequently, receive the Credibility Score from the blockchain node.
3. Decide if the Credibility Score is sufficient to trust the content.

The complete flow of verifying the Credibility Score is presented in Fig. 10.5.

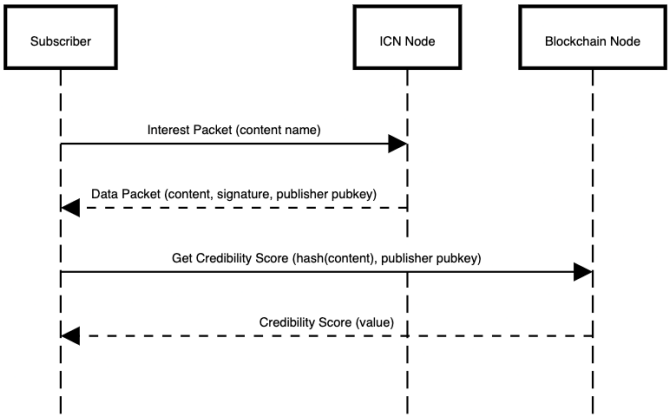


Figure 10.5. Flow of fetching and verifying content credibility score

The proof-of-time strength is calculated by counting the total number of transactions to the content hash address.

10.6. Blockchain structure

Blocks are fundamental primitives of every blockchain system. A typical block consists of a set of transactions, a Merkle Root [24] that represents the hash of all the transactions, a UNIX timestamp, a hash of the previous block (pointer), a hash of the whole block, and other

optional fields[2]. Blocks are created at some intervals; Bitcoin is designed to produce a new block every 10 minutes on average, Ethereum every 15 seconds, and Stellar every 5 seconds—this value is contractual[3]. We leverage this feature to achieve a global clock, whereby every block can be interpreted as a clock ”tick”. By counting the number of ”ticks” and multiplying by the ”tick” interval, we get an amount of proof-of-time—i.e., a Credibility Score. In Fig. 10.6, we show an example of three blocks containing proof-of-time claims. We assume that blocks are created with

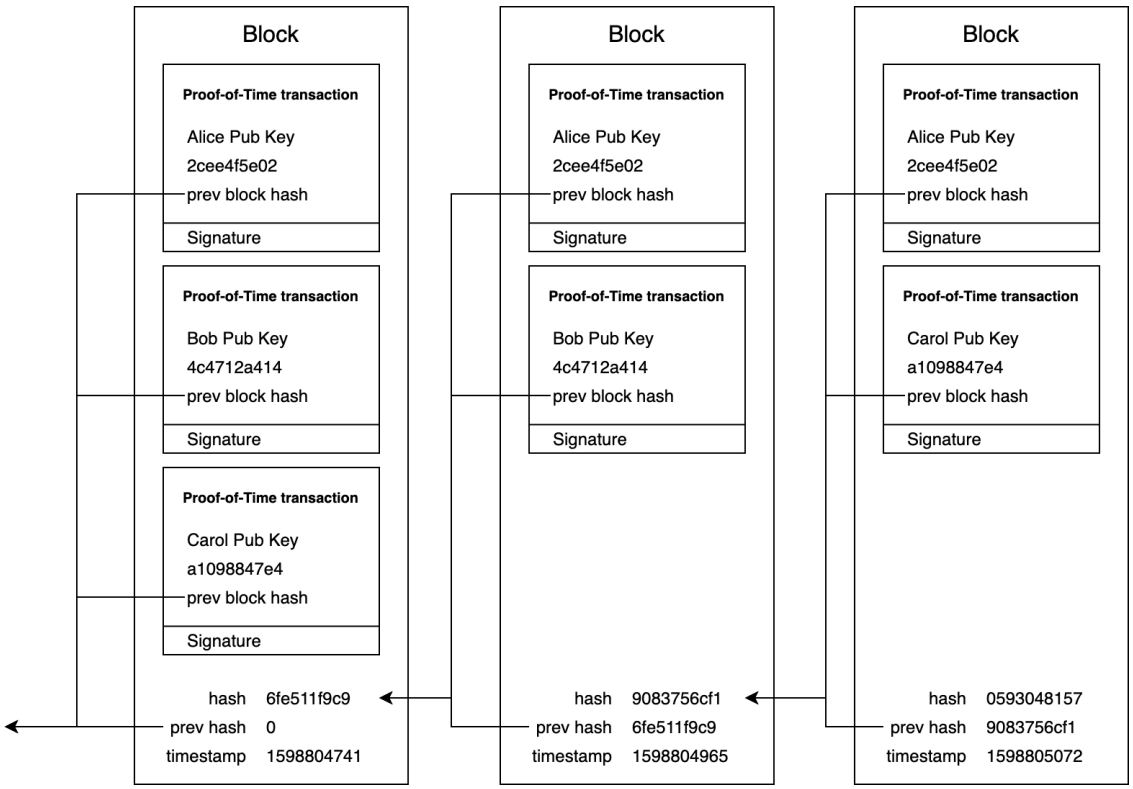


Figure 10.6. Blockchain of claims

10-minute intervals, and each content requires a Credibility Score of minimum 20 minutes. Alice first publishes her content to the ICN node getting the hash of the NDO, as shown in Fig. 10.4, then the proof-of-time transaction is created and published to the blockchain node. After 10 minutes, she again publishes the proof-of-time transaction, and after another 10 minutes she repeats the process. Three publications in a row certify that Alice has had Alice’s credentials for at least 20 minutes. Bob was able to publish only two claims, which we consider not enough to trust the content. On the other hand, Carol skipped the second block, which is considered a break in the proof-of-time chain; therefore, counting the proof-of-time for her content starts from the third block.

10.7. Algorithm

The algorithm used to process new blocks and checking content credibility score is presented in Listing. 10.1. Line 1 defines the state of finalized proof-of-time chains. We allow each content to be authenticated by multiple publishers. Each subscriber can decide which content

[2]Blockchains using proof-of-work consensus algorithms also include a nonce field—a proof of the proof-of-work puzzle

[3]Shorter block creation intervals give faster confirmations, but also speed up the blockchain size growth

```
1  stored_claims      # { [content-hash] : [publisher_pubkey] : integer }
2  candidate_claims  # { [content-hash] : [publisher_pubkey] : integer }
3  previous_block_hash
4
5
6  def process_new_block(block):
7      # handle broken proof-of-time chains
8      for content_hash in candidate_claims:
9          for publisher_pubkey in candidate_claims[content_hash]:
10             if (content_hash, publisher_pubkey) not in block.transactions:
11                 # proofs-of-time chain has been broken, store the length
12                 stored_claims[content_hash][publisher_pubkey] =
13                     candidate_claims[content_hash][publisher_pubkey]
14                 delete candidate_claims[content_hash][publisher_pubkey]
15
16     for transaction in block.transactions:
17         content_hash = transaction.hash
18         publisher_publickey = transaction.publisher_publickey
19         previous_block_hash = transaction.previous_block_hash
20         # check the previous_block_hash to prevent pre-signed proofs
21         if previous_block_hash != previous_block_hash:
22             continue
23         candidate_claims[content_hash][publisher_publickey] += 1
24
25     previous_block_hash = block.hash
26
27
28 def get_credibility_score(content):
29     if stored_claims[content] > candidate_claims[content]:
30         return stored_claims[content]
31     else:
32         return candidate_claims[content]
```

Listing 10.1. Processing a new block

publisher to trust. Line 2 defines the state of candidate claims—those that have not broken the proofs chain yet. Each blockchain node, upon receiving a new block, executes the method in line 6; first, it checks if some of the proofs have been broken (are not present in the block), and if so, they are finalized and stored in `stored_claims` (lines 10-14). Then it iterates over transactions present in the block, checks if the `previous_block_hash` matches the actual previous block hash (line 21), then extends the Credibility Score (line 23), and stores the current block hash as a `previous_block_hash`.

10.8. Federated Byzantine Agreement

Obviously, the proof-of-work consensus protocol (or any other so-called Nakamoto Consensus protocol, where the leader capable of updating the blockchain is elected in a lottery with the chance of winning depends on the amount of spent resource) is not suitable for the Stolen Credentials problem. However, consensus algorithms in distributed systems have been studied for decades, so there are many protocols to choose from.

We find Federated Byzantine Agreement (FBA)—and its blockchain implementation called Stellar Consensus Protocol (SCP)[2] the most suitable for our needs. In contrast to proof-of-work, where computational power dictates the contribution to consensus, FBA is based on a

trust model. It becomes a fast, lightweight, and asymptotic resistant[4]. Instead, the contribution value is determined by the node trustworthiness, similarly to the GI protocol. A new node joining the network has no contribution to the protocol until someone trusted starts to trust it.

Blockchain, like any other asynchronous distributed system, faces the FLP impossibility trilemma [25], where only two of three properties: Fault Tolerance, Liveness, and Safety, can be achieved. Most systems must be fault tolerant, so the choice is left between Liveness and Safety. Safety guarantees state consistency across all nodes in the network–if nodes do not agree on some transaction, they will not split into two different states, but rather wait until the conflict is resolved. Liveness guarantees that the consensus will always terminate, and the system will always be available to admit new transactions. When a conflict occurs, the ledger is split into two different versions until the conflict is resolved, but in the meantime it can process new transactions. Most of the blockchain protocols choose Liveness, i.e., tolerate temporary partitioning. It is argued that the time of the partitioning is short enough so that users expecting high credibility of the transaction can just wait until the chance of shifting the state is acceptably small[5]. The conflict settlement is again dictated by the computational power. The state that becomes the ancestor for the next block is considered the winner—-in this way, the system can guarantee permanent availability, even with just one working node.

Stellar Consensus Protocol (SCP), on the other hand, chooses Safety over Liveness. Once the state has been approved, it cannot be changed. State gets approved when the quorum of the network agree on the proposed state. SCP is based on Practical Byzantine fault tolerance (PBFT) [26] and extends its functionality by allowing open membership, therefore promoting decentralization. In SCP, each node picks its trusted set of nodes called *quorum slice* (in which it is *ipso facto* a member). The *quorum slice* should be different for each node, but naturally some nodes are more trustworthy and so picked more often for the quorum slices. Transitive trust for all node's *quorum-slice* members, then forms quorum. For any two quorums, there must exist a *quorum-intersection* to prevent network partitioning.

In non-FBA systems, the majority of the nodes determine the consensus decision on some state proposal. Once the proposal gets accepted by a quorum (a majority of the nodes), the rest of the network can be certain that other proposals will fail, because other proposals cannot reach the quorum and because the nodes cannot change their proposals. As a result, the whole network converges to a final decision. In decentralized systems, where nodes can join and leave at will, the total number of nodes in the network is unknown a priori. Therefore, it is hard to determine a majority. Additionally, open systems cannot rely on quantitative majority because it would open them to Sybil attacks[6]. To solve this problem, FBA introduces a federated voting process that starts locally and expands until it reaches the whole network. The local quorums must overlap with at least one common node to convey the voting decisions across different quorums. This so-called quorum-intersection property guarantees that if one quorum agrees on some value V,

[4] That is, a node with large computational power does not gain any advantage in the consensus protocol

[5] In proof-of-work, the chance of changing the state of some block diminishes as the chain of blocks attached to this block lengthens

[6] In this attack, a single entity can join the network as many nodes that to the rest of the network look as independent units, therefore able to force specific decisions

the other quorums cannot agree on non-V.

Federated voting starts when some node sends a broadcast to the network announcing a vote on a particular value V. By doing so, the node promises it will never vote against V. Each node sees how other nodes are voting by listening to their broadcass. If a node notices that some quorum of nodes voted on V then, by the definition of quorum, it can be certain that V will eventually be accepted by the whole network. Therefore, it can switch to an *accepting V* state and announce this to the whole network, just as it announced its vote. *accepting V* is stronger than voting–the latter means that the node will never vote for non-V while the former means that *each node in the network* will never accept non-V. When a node notices some quorum of nodes is *accepting V*, it *confirms V*, implying, by the definition of the quorum, that all nodes in the network will eventually have *confirms V*; this ends the process of federated voting.

The problem arises when nodes in a quorum intersection are Byzantine-failed, lying about the decisions made on each quorum. In a SCP whitepaper there is an assumption that the network is configured in such a way that even if the malicious nodes are removed from the network, the quorum-intersection property still holds, otherwise the network halts until the quorum-slices are reconfigured. We can only expect this property to hold in large real-world networks such as thhe Internet, which we are designing the protocol for.

Another problem with adopting FBA to our use case might be a DoS attack. A malicious publisher may publish a massive amount of proof-of-time transactions successfully, leading to network congestion. Stellar prevents that by introducing the transaction fees. Therefore, the attacker is discouraged by financial means. In our approach we do not want to introduce any payment mechanisms, so other defense mechanisms. DoS is a vital problem in ICN networking as such [27], hence we treat this topic as out-of-scope, referring to the generic solutions for ICN.

Among the various blockchain consensus protocols, not all are suitable as Internet-level protocols that have to run on resource-constrained network devices. If we consider IoT devices, we can leverage existing research, however, e.g., [28] suggests that Stellar consensus is not ideal for IoT devices since it is too slow. Yet in our case the proof-of-time claims are on order of minuter or hours rather tna milliseconds, thus we believe that our protocol is fast enough. Moreover, there already exists a proposal of a modified FBA algorithm using a virtual voting algorithm [29] that can achieve consensus with almost no communication overhead. We find this topic worthy of future work. For now we have settled on FBA in its raw form presented above.

11. COMPARISON

In comparing the two presented approaches, GI and Blockchain, we have focused on five main characteristics referred to as outcome, resilience, communication complexity, processing scalability (measured by the pieces of content the entire system can handle at a time), and fault-tolerance. One also has to consider the implementation requirements in terms of overlay IT infrastructure. We present the comparison in Table 11.1.

- Outcome – GI is indeterministic when it comes to a consensus decision. Even a high number of proof-of-times can sometimes result in an extinction, which is an unwanted property. Blockchain, on the other hand, is deterministic; a publisher with a high number of proof-of-time claims can be sure that his/her content will be authenticated. This is guaranteed by the safety property of FBA.
- Resilience – GI is sensitive to both chance, the network size and structure, and the ξ, and Z_{IA} parameters. While Blockchain—particularly using FBA—has just one assumption regarding the quorums-intersection property. We state that Blockchain is more resilient.
- Communication complexity – In GI, each node has to communicate with just a constant number of its neighbors, so the complexity is linear in the number of nodes—$O(|N|)$. In Blockchain, each node has to communicate with all other nodes in the network, so the communication complexity is quadratic in the number of nodes—$O(|N|^2)$. In this regard, GI prevails.
- Processing capability – to achieve safety, Blockchain has each network node process each piece of content sequentially, so the system is as fast as a single node in the network. Hence, adding nodes does not increase the total processing capability, which therefore is constant in the number of network nodes. In GI, communicates only with its neighbors. Therefore, nodes that are not connected can process different pieces of content independently–more network nodes increase the total processing capability, which is therefore linear in the number of network nodes.
- Fault-tolerance – GI in its raw form does not handle fail-stop, let alone Byzantine faults. Blockchain tolerates up to $\frac{|N|-1}{3}$ Byzantine nodes, similarly as Stellar FBA. Blockchain therefore outperforms GI in this regard.

Property	GI	Blockchain
Outcome	Indeterministic	Deterministic
Resilience	Depends on chance, $\|N\|$, ξ and Z_{IA} parameters, network structure (defensive alliance-sensitive)	Depends on network structure (quorums-intersection required)
Communication complexity	$O(\|N\|)$	$O(\|N\|^2)$
Processing capability	$O(\|N\|)$	$O(1)$
Fault-tolerance	None	Byzantine-fault tolerance up to $\frac{\|N\|-1}{3}$ faulty nodes
Required overlay infrastructure	NDN, PKI	NDN, PKI, and blockchain network

Table 11.1. Comparison of GI and Blockchain

12. SUMMARY

We have elaborated on the Stolen-Credentials Cpntent Poisoning Attack problem proposed in [1]. Assuming time- constrained operation of a potential malicious publisher, we have quantified the effectiveness of the existing graph infection-based GI protocol and proposed an alternative, blockchain-based approach. The presence of the time constraint allows to extend the authentication abstraction from *access to credentials* to both *access to credentials* and *access to time*, coining the term *Proof of Time*. We have developed simulators and analyzed the proof-of-time implementation based on GI. Among others, we have observed that GI struggles with the Defensive Alliance problem described in Section 8.2, and remarked on GI's sensitivity to the parameter configuration (ξ and Z_{ia}), which may be troublesome in Internet-level graph structures can change in time. Then we have generalize the considerations to a distributed consensus problem and proposed a solution based on Blockchain technology. Besides solving the consensus problem, it also offers valuable properties such as security, integrity, immutability, and Byzantine-fault tolerance. We have proposed a scheme in which the content authentication mechanism is embedded not in the consensus mechanism, but as an overlay structure (blockchain) on top of the consensus algorithm. In consequence, we gain a deterministic and reliable authentication mechanism called Credibility Score. We have found Stellar and its Federated Byzantine Agreement consensus protocol to be a blockchain implementation most suitable for our needs. One disadvantage of it is the quorum-intersection requirement described in Section 10.8. Additionally, the Blockchain solution is more demanding in terms of overlay infrastructure and does not scale well with the network size. We believe that authentication schemes based on proof-of-time may become a valuable option for the future development of ICN environments.

BIBLIOGRAPHY

[1] Jerzy Konorski. “Mitigating Time-Constrained Stolen-Credentials Content Poisoning in an NDN Setting”. In: *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE. 2019, pp. 1–7.

[2] David Mazieres. *The stellar consensus protocol: A federated model for internet-level consensus.(2015)*. URL: https://www. stellar. org/papers/stellar-consensus-protocol. pdf, 2015.

[3] *(4) A New Way to look at Networking - YouTube*. https://www.youtube.com/watch?v=gqGEMQveoqg. (Accessed on 09/16/2020).

[4] *Named Data Networking (NDN) - A Future Internet Architecture*. https://named-data.net/. (Accessed on 09/16/2020).

[5] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, Rebecca L Braynard. “Networking named content”. In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. 2009, pp. 1–12.

[6] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, Borje Ohlman. *A survey of information-centric networking*. IEEE Communications Magazine, vol. 50, nr 7, 2012, pp. 26–36.

[7] Cesar Ghali, Gene Tsudik, Ersin Uzun, et al. “Needle in a haystack: Mitigating content poisoning in named-data networking”. In: *Proceedings of NDSS Workshop on Security of Emerging Networking Technologies (SENT)*. 2014.

[8] Yong Yu, Yannan Li, Xiaojiang Du, Ruonan Chen, Bo Yang. *Content Protection in Named Data Networking: Challenges and Potential Solutions*. 2018. arXiv: 1810.11179 [cs.CR].

[9] Tan Nguyen, Xavier Marchal, Guillaume Doyen, Thibault Cholez, Rémi Cogranne. “Content poisoning in named data networking: Comprehensive characterization of real deployment”. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 72–80.

[10] Petros Maniatis, David SH Rosenthal, Mema Roussopoulos, Mary Baker, Thomas J Giuli, Yanto Muliadi. “Preserving peer replicas by rate-limited sampled voting”. In: *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 2003, pp. 44–59.

[11] Xinyi Zhou, Reza Zafarani. *Fake news: A survey of research, detection methods, and opportunities*. arXiv preprint arXiv:1812.00315, 2018.

[12] Brian J Coburn, Bradley G Wagner, Sally Blower. *Modeling influenza epidemics and pandemics: insights into the future of swine flu (H1N1)*. BMC medicine, vol. 7, nr 1, 2009, p. 30.

[13] Qun Li, Xuhua Guan, Peng Wu, Xiaoye Wang, Lei Zhou, Yeqing Tong, Ruiqi Ren, Kathy SM Leung, Eric HY Lau, Jessica Y Wong, et al. *Early transmission dynamics in Wuhan, China, of novel coronavirus–infected pneumonia*. New England Journal of Medicine, 2020.

[14] Mark Channels Read. *EID: High Contagiousness and Rapid Spread of Severe Acute Respiratory Syndrome Coronavirus 2*.

[15] Yuval Noah Harari. *Sapiens: A brief history of humankind*. Random House, 2014.

[16] *RFC 4880 - OpenPGP Message Format*. https://tools.ietf.org/html/rfc4880. (Accessed on 09/25/2020).

[17] *2020 Data Breach Investigations Report: Official | Verizon Enterprise Solutions*. https://enterprise.verizon.com/resources/reports/dbir/. (Accessed on 09/17/2020).

[18] *RFC 6819 - OAuth 2.0 Threat Model and Security Considerations*. https://tools.ietf.org/html/rfc6819. (Accessed on 09/04/2020).

[19] Kahina Ouazine, Hachem Slimani, Abdelkamel Tari. *Alliances in graphs: Parameters, properties and applications—A survey*. AKCE International Journal of Graphs and Combinatorics, vol. 15, nr 2, 2018, pp. 115–154.

[20] Leslie Lamport, Robert Shostak, Marshall Pease. "The Byzantine generals problem". In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 203–226.

[21] Leslie Lamport. *Proving the correctness of multiprocess programs*. IEEE transactions on software engineering, 1977, pp. 125–143.

[22] Satoshi Nakamoto, A Bitcoin. *A peer-to-peer electronic cash system*. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf, 2008.

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

[24] Ralph C Merkle. "A certified digital signature". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 218–238.

[25] Michael J Fischer, Nancy A Lynch, Michael S Paterson. *Impossibility of distributed consensus with one faulty process*. Journal of the ACM (JACM), vol. 32, nr 2, 1985, pp. 374–382.

[26] Miguel Castro, Barbara Liskov, et al. "Practical Byzantine fault tolerance". In: *OSDI*. Vol. 99. 1999. 1999, pp. 173–186.

[27] Paolo Gasti, Gene Tsudik, Ersin Uzun, Lixia Zhang. "DoS and DDoS in named data networking". In: *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2013, pp. 1–7.

[28] Mehrdad Salimitari, Mainak Chatterjee. *A survey on consensus protocols in blockchain for iot networks*. arXiv preprint arXiv:1809.05613, 2018.

[29] Naïri Usher Sean Rowan. *The Flare Consensus Protocol: Fair, Fast Federated Byzantine Agreement Consensus*. https://static1.squarespace.com/static/5d9538470c5b607d9c8b8953/t/5d953a5a8bb63364633a0b42/1570060892254/FCP.pdf. (Accessed on 08/28/2020).

LIST OF FIGURES

LIST OF TABLES

Praca dotyczy bezpieczeństwa informacji w sieciach informacjocentrycznych (ang. Information-Centric Networks, ICN) na przykładzie technologii dystrybucji treści pod nazwą Named Data Networking (NDN). Jednym z problemów jest atak zatruwania treści (ang. Content Posioning Attack, CPA), a zwłaszcza jego silniejsza forma—atak zatruwania treści przez wydawców posugujcych si kradzionymi danymi uwierzytelniajcymi (ang. Stolen-Credentials CPA). Dla czytelniejszego wprowadzenia wykonany został przegląd innych technologii zmagających się z tym problemem, w tym Wikipedii, LOCKSS, oraz technologii mediów społecznościowych. Te ostatnie pozwalają dostrzec podobieństwa między znanym problemem fałszywych wiadomości (ang. fake news) a zatruwaniem treści. Zaproponowane zostało rozwiązane oparte na ograniczonym czasie dostępu do fałszywych wiadomości. Następnie rozwiązanie to zostało uogólnione na metody uwierzytelniania treści, nie tylko w mediach społecznościowych, ale również w sieciach ICN. Uogólnione rozwiązanie, nazwane Dowód Czasu (ang. Proof-of-Time), polega na rozszerzeniu mechanizmu uwierzytelniania treści o składnik dostępu czasowego do uwierzytelnień pozwalających na publikacje danych. Następnie bazując na pracy [1] zbadany został algorytm infekcji na grafach w celu implementacji wcześniej założonego mechanizmu. Przebadane zostały różne modele używane w epidemiologii, które pozwalają na symulację oraz analizę procesów epidemiologicznych. Kolejno przebadane zostały struktury grafów oraz algorytmy generatorów pozwalające na budowę sieci o różnych właściwościach, m. in. o strukturze grafów przypadkowych i bezskalowych.

Kolejny rozdział poświęcony został przeglądowi ataków na dane uwierzytelniające. Informacja ta pozwala na oszacowanie wymaganego dostępu czasowego do uwierzytelnień w celu efektywnego działania mechanizmu Dowodu Czasu.

Następny rozdział szczegółowo opisuje zasadę działania algorytmu infekcji na grafach, w szczególności opisuje maszynę stanów, która steruje zachowaniem każdego węzła w sieci. Opisuje również mechanizm rozprzestrzeniania się infekcji.

Stworzone dla celów tej pracy symulatory pozwalają na dogłębną ilościową analizę algorytmu oraz na obserwacje rozmaitych anomalii. Jedną z nich jest zjawisko nazwane Sojuszem Defensywnym, który może skutecznie zablokować rozprzestrzenianie się epidemii—co w przypadku tego algorytmu jest efektem niepożądanym .

Następnie problem infekcji na grafach został uogólniony do problemu konsensusu. Zauważone został, że końcowy stan sieci – epidemia lub wygaśnięcie infekcji, może być interpretowany jako decyzja wypracowana na drodze konsensusu wszystkich węzłów sieci. Z tego powodu problem ten może być rozwiązany z użycien znanych algorytmów konsensusu w systemach rozproszonych.

Zaproponowane zostało rozwiązanie oparte o technologię blockchain, które poza rozwiązaniem problemu konsensusu, dostarcza wartościowych cech, takich jak odporność na błędy bizantyjskie (ang. Byzantine-fault tolerance) oraz uniezależnienie od topologi sieci. Dodatkowo rozwiązanie to pozwala na wprowadzenie progresywnego uwierzytelniania treści—w przeciwieństwie do algorytmu infekcji na grafach, gdzie dana treść mogła mieć jedynie status binarny–uwierzytelniona albo nieuwierzytelniona—w rozwiązaniu opartym na technologii blockchain możliwe jest elasty-

czne interpretowanie wiarygodności każdej treści z osobna. Treści typu muzyka, filmy oraz zdjęcia są mniej wrażliwe na ataki zatruwania treści, niż treści takie jak oprogramowanie, lub strony internetowe wymagające podania poufnych danych. System oparty technologii blockchain rejestruje długość Dowodu Czasu, pozostawiając jego interpretację końcowemu użytkownikowi.

Do dalszych badań wybrany został protokół konsensusu Federated Byzantine Agreement (FBA) ze względu na możliwość zabezpieczenia otwartych sieci zaufania (których węzły mogą w każdej chwili dołączać lub odłączać się), szybkość działania oraz tolerancję obecności do jednej trzeciej węzłów wykazujących zachowania bizantyjskie. FBA wymaga, aby każde sformowane kworum (podzbiór węzłów) miało część wspólną z każdym innym kworum; niespełnienie tego warunku prowadzi do utraty spójności stanu sieci. Dodatkowo rozwiązanie oparte na technologii blockchain cechuje się większą komplikacją i gorszą skalowalnością wraz z liczbą węzłów sieci, a także wprowadza dodatkowe wymagania co do infrastruktury nakładkowej.

Na koniec pracy przedstawione zostaje porównanie obu podejść. Algorytm infekcji na grafach jest szybszy i prostszy, jednak algorytm oparty na technologii blockchain cechuje większy determinizm, odporność na błędy oraz elastyczność.