Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Privacy implications of AMQ-based PQ TLS authentication

Semester Project

Dimitri Francolla

March 29, 2024

Advisors: Prof. Dr. Kenny Paterson, Mia Filić, Shannon Veitch

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

Post-quantum digital signature schemes generate significantly larger signature sizes compared to traditional ones, drastically increasing the size of certificate chains used in TLS for entity authentication. This introduces a significant performance bottleneck due to multiple round-trips needed by the underlying TCP connection to transmit all the authentication data. Recent solutions have proposed an Approximate Membership Query (AMQ) based approach for Intermediate Certificate (ICA) certificate suppression. This approach has been shown to drastically reduce the communication overhead for authentication data, avoiding multiple round-trips. On the other hand, it requires the client to reveal information about its known ICAs to the other entity. Over 95% of Google's traffic is performed through the TLS-secured HTTPS protocol, meaning that the near totality of online communications would be affected by this information leakage. We investigate the privacy implications of such an approach, by concretely evaluating how much information could be leaked to a malicious adversary impersonating a TLS server. To do so, we formally define the privacy leakage and the risk for the user in this scenario. We perform an exploratory analysis on the current state of certificate-based PKI, particularly focusing on the distribution of ICAs. We define an adversarial model and a set of experiments to concretely evaluate the privacy leakage under the outlined assumptions. We finally discuss some mitigation strategies and possible modifications to improve the performance of the ICA suppression method.

Contents

Chapter 1

Introduction

Transport Layer Security (TLS) is a protocol for secure communication used in most online applications. It runs on the Transmission Control Protocol (TCP), and it is made up of two sub-protocols: the *handshake protocol* and the *record protocol*. The handshake protocol serves the purpose of establishing all the relevant *shared secrets* in order to later provide a secure channel, as well as enabling *peer authentication* between the two parties by using public key cryptography. During the handshake protocol, peers also negotiate the cipher suites to employ during the rest of the communication. The record protocol on the other hand uses all the parameters established during the handshake to provide a secure tunnel between the two parties, guaranteeing connection privacy and message integrity.

The current Internet infrastructure relies heavily on TLS connections between clients and servers in order for the users to correctly access online content: more than 95% of Google's traffic is performed through the TLS-secured HTTPS protocol [2], and other studies suggest that there is a general wide adoption of HTTPS for all kinds of web traffic [4]. Consequently, the delay introduced by the establishment of a TLS connection plays a crucial role in the usability and overall experience of a user surfing the web.

With the continuous improvements in the field of quantum computing, the "harvest now, decrypt later" type of attacks are becoming increasingly worrisome for the cryptographic world. These attacks consist of capturing large amounts of unreadable encrypted traffic and storing it until new breakthroughs allow for efficient decryption of the data. Large enough stable quantum computers can in fact potentially break current traditional cryptographic primitives [22], [18], thus representing a threat for most of the services used online, no matter how distant in the future they could be. Such an outcome could have a global impact, causing consequences in almost every aspect of online communication: instant messaging services would not be able to properly protect the user's data, password managers would not be able to securely store credentials, and governmental and military sensitive information could be revealed. Consequently, privacy and anonymity on the Internet would not be attainable anymore. This creates the need for robust quantum-resistant, or post-quantum (PQ), cryptographic primitives. NIST has already started a standardization process to select future PQ alternatives both for encryption and authentication [14].

On the other hand, the adoption of PQ signature schemes in the TLS 1.3 handshake protocol introduces a significant performance bottleneck, which will in most cases render the introduced delay unacceptable for many TLS applications. An acceptable TLS handshake delay is usually considered to be within the range of a few up to a few tens of milliseconds, heavily depending on the kind of application and how time-sensitive it is. A delay of around 100 milliseconds already starts becoming noticeable by users, for example when loading an Internet webpage. The results of Sikeridis et al. in [24] show that almost all configurations of the PQ algorithms selected by NIST for future standardisation introduce a delay of 50 to 500 milliseconds at a minimum, except for the lowest security levels of Dilithium [6] and Falcon [9] which manage to stay between 9 and 15 milliseconds.

These newly proposed signature schemes are quantum resistant, but on the other hand, they require much larger key and signature sizes. The result is a drastic increase in the size of the messages exchanged during the TLS handshake. Entity authentication in TLS is performed by exchanging signed certificate chains, which the receiver verifies to authenticate the sender. Certificate chains include a certificate of a trusted root Certificate Authority (CA), the end-entity's own certificate, also called *leaf certificate*. They can also potentially include certificates belonging to Intermediate Certificate Authorities (ICAs). Root CAs can delegate to them the ability to issue other certificates to end entities or even other ICAs. This means that certificate chains contain *at least* two certificates, but they can be more than that due to the presence of ICAs. The adoption of PQ signatures would drastically increase certificate signature sizes. If the certificate chain data exceeds TCP's congestion window size, it needs multiple round trip times to be fully transmitted.

To overcome the communication overhead introduced by PQ solutions, and in general, to optimize the handshake protocol as much as possible, there have been various proposals. One such technique is Intermediate Certificate Authority (ICA) certificate suppression [11]: it consists of the omission of the signatures for ICAs which are already known to both parties to reduce the size of the messages exchanged during the handshake. This approach could prove to be extremely useful in mitigating the detrimental increase in authentication data that PQ signature schemes introduce. In particular, Sikeridis et al. [23] propose to use an Approximate Membership Query Probabilistic Data Structure (AMQ-PDS) to advertise the known certificates by one party to the other, so that unnecessary signatures can be omitted from the authentication message. This solution achieves excellent results in terms of the reduction of the authentication data. However, it also forces the client to reveal the server information about its stored ICAs. The ICAs stored by the client are collected during its Internet navigation, by visiting different web pages. This kind of information can potentially be used by a malicious server to reconstruct the victim's visited domains, hindering the user's privacy. With this work, we study and analyse this particular instance, and evaluate the severity of the privacy leakage.

The contributions of this project start with outlining the privacy concerns we identified in the solution proposed by Sikeridis et al. [23]. We formally define a notion of privacy leakage and show the extent of it by running experiments on real data. We perform an exploratory analysis of the Public Key Infrastructure (PKI) state in general, and more specifically focus on ICAs in the wild to have a clearer picture of the real scenario. Then we design a model for the analysed scenario: this includes a client and an adversary model. We specify their capabilities, limitations and modes of interaction. We also present our experimental setup, giving specific details on how we conducted our experiments. More specifically, we design some possible attack protocols for the adversary and we outline the assumptions we make on the two parties. Finally, we present the obtained results. We also discuss possible improvements regarding the proposed ICA suppression method and present possible further investigations that remain open at the end of our project.

The results we obtain reveal that the designed attacks on the privacy of the client have an extremely variable efficacy, depending on the assumptions that the adversary can make about the client. More specifically, our results show that to achieve a significant level of effectiveness, the adversary needs to have some kind of additional data about the client, that enables the attacker to restrict the scope of the attack. If this condition is satisfied, the adversary can potentially infer a domain that the client has visited with $\approx 30\%$ confidence level in the best-case scenario.

By themselves, the designed attacks against the AMQ-based ICA suppression method are relatively weak. Despite this fact, they could prove to be a powerful tool at the disposal of real-world attackers with access to many other fingerprinting techniques that can be used against the client. The proposed attacks on the ICA suppression method could be used in pair with these other fingerprinting techniques to further refine their results.

Chapter 2

Background

2.1 PKI and ICAs

Public Key Infrastructure (PKI) is one of the main pillars that enable the Internet to properly and smoothly function as it currently does. PKI binds the identities of the various entities that compose the Internet together with their public keys, facilitating authentication processes between parties during a connection. This binding is mainly done through the use of digital certificates, which are issued and signed by Certificate Authorities (CA) with their private key, which function as roots of trust for the entire ecosystem. Each entity that wants to own a digital certificate must register through a CA, which will in turn issue a certificate for that entity, binding it to its public key. The issued certificate can later be used for authentication and communication purposes, for example during a TLS connection.

To facilitate and decentralize the certificate issuance process, CAs can also delegate some tasks to other entities, enabling them to issue digital certificates. These entities to which root CAs delegate certificate issuance are called Intermediate CAs (ICAs). This hierarchical structure provides performance benefits such as improved scalability and ease of management. It allows for a more granular control over certificate issuance: for instance, ICAs can issue certificates for specific geographic regions, organizational units, or purposes. Most importantly, decentralising the issuance process grants enhanced security for the entire infrastructure. Segregating roles and responsibilities among different ICAs helps mitigate risks arising from a compromised entity within the infrastructure. If an ICA is compromised, the consequences are restricted to a subset of all the issued certificates, a much narrower scope compared to an instance of root CA compromise.

By using this method, entities must provide a certificate chain rather than a single end-point certificate when authenticating their identity and key material. Certificate chains represent the trust relationship between the end entity and a trusted root CA. When a new certificate is issued, either for an ICA or an end entity, it is signed by the issuing entity. Only root certificates are self-signed by the trusted authority. This process forms a chain of trust, starting from the end entity certificate to the trust anchor which is the root CA. Each certificate also includes a reference to its issuer's certificate, to enable the verification of the certificate chain.

Certificate signatures can be verified using the public key contained in the issuer's certificate, while root CA certificates are self-signed. The verification of the certificate chain consists of verifying *all* signatures up to and including the root certificate. If the certificate signature verification under the claimed issuer's public key succeeds for each certificate in the chain, then the entire chain is verified. Equivalently, the trust relationship between the specified trusted root CA and the end entity is verified, and thus the end entity can be considered correctly authenticated.

We can then visualize the certificate-based PKI ecosystem as a forest of different trees. They are rooted in root CA certificates, which branch off into ICAs. ICAs can either delegate end entity certificate issuance to other ICAs or do it directly themselves. Edges in this forest represent the parent node issuing and signing the child node's certificate. In this way, there can even be chains with two or more ICAs. Finally, the leaves of these trees are represented by end entity certificates, which could belong to web servers, for example.

2.2 PQ and TLS

Let us now focus on the TLS 1.3 protocol [19]. It consists of two subprotocols: the handshake protocol and the record protocol. The handshake protocol makes use of asymmetric cryptography and PKI to authenticate one or both parties, to negotiate the cryptographic algorithms to use and to establish some shared ephemeral key material. This key material is later used in the record protocol to create a secure channel, in which the actual communication between the two parties takes place. In particular, the server's (and optionally the client's) certificate chain is sent during the handshake protocol, together with their respective signatures, as part of the entity authentication protocol.

The need for standardization and introduction of PQ cryptographic primitives, including digital signatures, into the TLS ecosystem [25] introduces a problem: PQ digital signature schemes offer a quantum-safe alternative to their traditional counterparts but are also much more expensive in terms of signature size. Using PQ digital signature schemes significantly increases the size of the certificate signatures that entities must send during the handshake to allow entity authentication. This can lead to a substantial performance bottleneck: the underlying TCP connection might need multiple round-trips to transmit all the authentication data, resulting in very large connection delays. Initial measurements obtained on the signature schemes selected by NIST for future standardization yielded quite negative results. Virtually all configurations of such schemes result in the introduction of unacceptable delays for TLS-based applications, and by quite a big margin [24].

2.3 Intermediate Certificate Suppression

Certificate chains enable entities in peer-to-peer communication such as in TLS to authenticate one another by using asymmetric cryptography and verification of digital signatures. Whenever an entity needs to authenticate itself, it needs to send the entire certificate chain together with the respective certificate signatures rather than a single digital certificate. This introduces both a communication overhead, because of the signatures that need to be sent by the authenticating party, but also a computational overhead, because the signatures need to be individually verified.

In most cases, verification algorithms for the currently used digital signature schemes are extremely fast. Thus, the computation time needed does not significantly impact the overall execution time for entity authentication in TLS 1.3. On the other hand, the transmission of the certificate chain with the respective signatures is something that could introduce some delay in the communication. Naturally, the transmission time is proportional to the size of the chain and signatures to be transmitted. This fact becomes even more crucial in TLS-based applications on the Internet, where the user experience is heavily impacted by the connection delay, which dictates for example the speed at which a certain webpage is loaded and displayed to the user. At the current state, traditional digital signatures do not introduce an excessive communication overhead. This could change in the future with the introduction of quantum-resistant signature schemes, which produce significantly larger signature sizes.

One solution that has been proposed to reduce the size of the certificate chain to be sent is called ICA Suppression. It consists of the omission of ICA signatures in the chain. In the best-case scenario, only the end-entity certificate should be sent to the other party to minimise the overhead introduced by the authentication process, even in the case of PQ signature schemes.

The omission of ICAs from the certificate chain is possible only if the two parties know which ICAs are already known by the other. If one party omits an ICA, but the other does not already know the omitted certificate, it will not be able to successfully perform the authentication process. For this reason, entity authentication with ICA suppression requires a process through which parties can learn which ICAs can be omitted and which ones cannot.

For example, preloading a common database of ICAs into software products constitutes a possible solution to ensure common knowledge of omitted ICAs. It embeds a shared set of universally known ICAs into every entity that uses the same software product. Mozilla set up an ICA Preloading Framework [27] consisting of a list containing around 1400 ICAs which is preloaded into Firefox clients.

Another possible solution to ensure common ICA knowledge consists of the two parties explicitly communicating to each other which ICAs they already know. This way, the other party can omit the ICAs known to the other party from their certificate chain. This approach can result in diminishing returns if not performed in a space-efficient way. We will focus on one proposed method specifically, which will be presented in Chapter 3.

2.4 AMQ-PDS

Probabilistic Data Structures (PDS) are a class of data structures that makes use of probabilistic algorithms to provide approximate answers to queries about the dataset they contain. Their big advantage over the usual deterministic data structures is that they are designed to handle very large amounts of data while ensuring extremely good space and time efficiency. Their downside is represented by their probabilistic nature. It introduces a (typically) small but always present error rate. This way, where the setting permits it, it is possible to trade off some accuracy in exchange for quick handling of large amounts of data with good space efficiency.

Approximate Membership Query (AMQ) filters are a set of space-efficient probabilistic data structures which support approximate membership query operations. These queries answer whether a certain element is contained in the filter or not. On the downside, they come with a false positive rate ϵ. False negatives cannot occur. This means that a successful approximate membership query indicates that an element is in the set with high probability, i.e., with probability $1 - \epsilon$. On the other hand, if an element was never inserted, the query will result in a false positive with probability ϵ.

The most famous AMQ-PDS is the Bloom Filter, designed in 1970. It is by far the most used and adopted in practice, but it comes with some inconveniences, in particular the lack of element removal from the set. Nonetheless, they are often applied in the context of network security [3] and they have a quite wide industry adoption despite their downsides and the presence of arguably better alternatives.

For example, Cuckoo Filters [7], which we will focus on for the sake of this project, are a newer design, from 2014. Cuckoo Filters support the deletion of elements and they have been shown to perform generally better than other Bloom Filters' extensions which support deletion.

2.5 Related Work

There are various works which recognize the need and try to implement some solutions for post-quantum primitives adoption into the TLS ecosystem. In this IETF RFC draft [25] for a TLS Hybrid Key Exchange, Stebila et al. propose an hybrid TLS for post-quantum transition. It makes use of two or more cryptographic suites. The security of the TLS session is guaranteed as long as at least one of the employed component algorithms is secure (i.e., has not been cryptographically broken). This enables early adopters of post-quantum algorithms to benefit from PQ security while still being guaranteed at least the level of security provided by traditional algorithms.

Multiple works have been focused on the problem of the authentication data size in PQ TLS and how to solve it. For example, [21] propose a TLS 1.3 alternative, called KEMTLS, which, as they show, removes the need of PQ signature schemes. This approach results in improved space efficiency and greatly reduces the communication overhead. Another solution is proposed in [12], which firstly quantifies the slow-down of TLS handshakes when dealing with heavy authentication data, and secondly proposes an ICA suppression framework which consists of a flag for signalling the use of the mechanism, and then caching techniques to actually store the known ICAs.

Finally, in this work, we will mainly focus on the proposal by Sikeridis et al. [23], in which they design an AMQ-based solution for ICA suppression. Entities should keep track of their known ICAs using a Cuckoo filter to store them. Then, during the TLS 1.3 handshake, they can send their filter to the other party. The other entity performs on approximate membership query for every intermediate certificate in its chain. If successful, that ICA and its signature can be omitted from the chain. Due to the probabilistic nature of Cuckoo filters, there is an ϵ false positive probability, which would lead one party to omit an ICA which in reality the other peer does not know. In that case, the authentication would fail and the system would fall back to the standard PQ TLS 1.3 handshake. The benefit is that the modified handshake is extremely efficient and can be performed in a significantly shorter time than the normal PQ TLS handshake.

Chapter 3

AMQ-based ICA suppression

In this chapter, we present the implementation details of the AMQ-based ICA suppression method. More specifically, we focus on the exchange of the ICA filter from the client to the server, as this is the relevant part of the privacy discussion. We will not provide details on performance and deployment, since it is not relevant to our goals.

We then present the privacy concerns that arise from the proposed method. A malicious server that receives the client's filter has access to the user's known ICAs. The ICAs which a user knows are directly linked to the websites it has visited. Consequently, this information leakage could enable the adversary to reconstruct the victim's navigation destinations, violating the user's privacy.

3.1 Overview

Figure 3.1 graphically depicts the part of the proposed protocol that is relevant for privacy leakage discussion. Consider a scenario in which the client authenticates the server. The protocol runs as follows.

The client's filter can be continuously updated with the addition of newly discovered ICAs, and at the same time revoked ones can be removed, thanks to the use of filters that enable dynamic updates such as Cuckoo Filters [7] or Quotient Filters [16]. When the client initiates a new connection, it includes its filter in its first handshake message to the server.

Upon receiving the filter, the server can locally check whether the intermediate certificates in its certificate chain are known to the client. This is done by performing membership queries on the filter for each intermediate certificate. If they are already known to the client, the server can omit their signatures from the certificate chain sent in response to the client as part of the authentication process. Otherwise, the server will simply fall back to

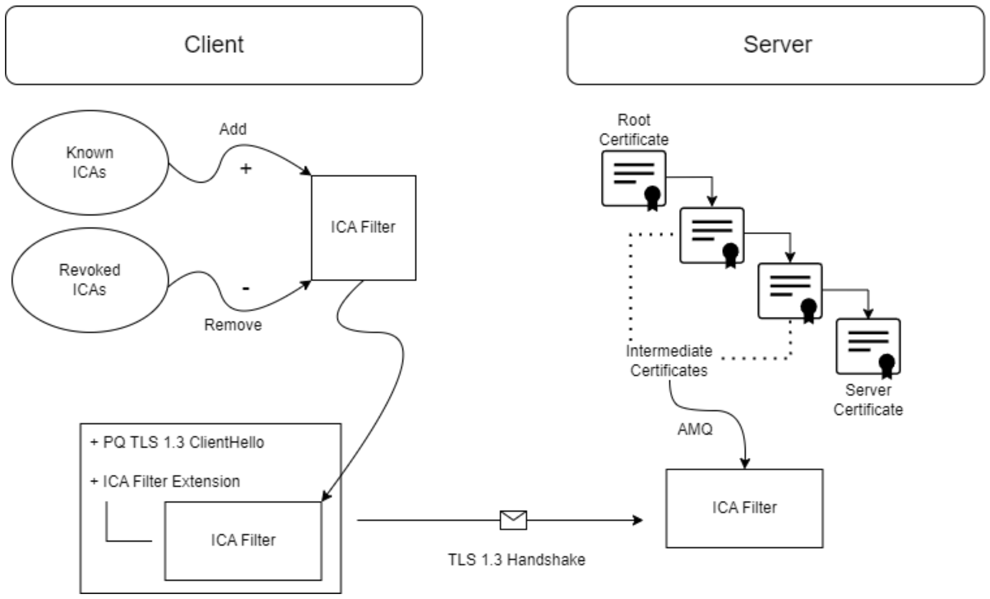

Figure 3.1: Diagram that describes the proposed AMQ-based handshake, specifically showing the addition of the ICA filter in the ClientHello message, and how it is later used by the server.

the usual authentication protocol and include the full certificate chain in its response. The filter could also return a false positive with probability ϵ, in which case the server will omit that specific ICA from the certificate chain even if the client does not already know it. In this instance, the server's verification will fail, and the client will start a new TLS 1.3 connection, this time omitting the filter. The remainder of the handshake will follow the normal PQ TLS 1.3 protocol. For this reason, the parameters of the filter should be tuned such that the false positive rate is small.

The solution for AMQ-based ICA suppression proposed by Sikeridis et al. [23] has a concrete performance impact on the PQ TLS 1.3 handshake, as shown in the results of the original paper. In fact, it can lead to an average decrease by 73% of the exchanged authentication data. Nonetheless, it also entails the exchange of new data from the client to the server, in the form of the ICA filter, which is normally not included in the usual TLS connection.

The initial message in the TLS 1.3 handshake, which is the one that would contain the client's ICA filter, is sent as cleartext, not effectively protected by any kind of encryption mechanism. This traffic can be passively observed and recorded by any on-path adversary. To overcome this risk, the authors suggest encrypting the first handshake message, as described in the IETF draft [20].

The use of encryption would mitigate the risk of passive attacks on the privacy of the client. Nonetheless, there is still one open problem that is not fully addressed by this method: a scenario in which the server is under the control of a malicious adversary. The adversary could either directly own the server or could have corrupted it and gained read access to it. For this reason, it is crucial to closely investigate any possible consequences that could arise from such a modification to the standard protocol.

3.2 Privacy Concerns and Leakage

In this instance, we want to specifically focus on the fact that the client provides the server with some additional information that is normally not available when performing the usual PQ TLS 1.3 handshake. We want to understand what could be the implications and consequences of this information leakage.

Evaluating the impact of such leakage is generally hard, as privacy is not an intrinsic quantitative feature of a cryptographic system. It also generally has many different connotations depending on the context in which it is intended. For this reason it is crucial to clearly define the type of scenario that we want to investigate, and who are the entities involved in it. We are able to do so by clearly defining an adversarial model and goal. This includes the adversarial capabilities and the assumptions that we make about the two entities involved.

More specifically, this work focuses on the implications of additional information leakage for a regular Internet user. By considering this scenario, we can realistically evaluate if any attack designed against the protocol can have repercussions on a broader scale, in case of wide adoption of the ICA suppression mechanism. This also enables us to have minimal assumptions on the type of traffic that a simulated client would need to generate, which makes testing much easier. For instance, there are many well-known databases which record top-sites rankings based on popularity and traffic. These rankings provide a concrete representation of the most common navigation destinations for the vast majority of Internet users. More on this will follow in Section 4.1.

When working on this project we had one specific representative example in mind for the kind of scenario we want to investigate. Consider a user who is not particularly educated and knowledgeable in technology and the practical underpinnings of Internet browsing. Our work addresses the following questions for such a user: how, and to what extent can this user be affected by the privacy consequences entailed by AMQ-based ICA suppression? How can an attacker take advantage of the additional information provided by the ICA filter, and how can that be concretely used against the user?

Since the filter only contains information about intermediate certificates known by the client, we imagine on a high level what could be useful for an adversary to extract from the client's ICA data. For instance, imagine the adversary can retrieve a set of domains visited by the client with high confidence. Imagine also that the server controlled by the adversary requires a login, and consequently reveals the client's email. With the new information about the client's visited domains, an attacker could craft much more sophisticated and targeted phishing e-mails, which would look and feel authentic and legitimate to the victim. If the filter could reveal information about the visited domains of the client such as what e-banking portal the client connected to, or the client's mobile carrier, this kind of privacy leakage could open up many possibilities for an attacker: targeted phishing could be performed in order to steal access credentials to the victim's e-banking, or it could even lead to SIM swapping [13] in order to access other accounts of the victim.

In addition, it is not particularly difficult for attackers to potentially access user's ICA filters: if the message containing the filter is not encrypted, then the adversary can simply passively record it by eavesdropping on the client-server connection. Even with the use of public key encryption to protect the message, the adversary can force users to connect to one of its servers. It is not unreasonable to assume that attackers could potentially force users to connect to malicious servers under their control to get access to this kind of information.

For this reason, we believe it is crucial to further investigate the privacy implications of this method and to understand whether it is feasible for an adversary to extract a set of domains visited by the victim, and if so, with what level of accuracy this can be done. The attacks we designed will be presented in detail in Chapter 5.

Chapter 4

ICAs in the Wild

In order to understand what a malicious adversary could extract from the information about the client's known ICAs it is imperative to have a clear understanding about the web PKI ecosystem based on certificates and its current state.

More specifically, we want to focus on understanding how common ICAs are in the wild and how many we can expect to see on average in certificate chains. This information is extremely useful for AMQ-based ICA suppression since it allows to estimate the values of important implementation parameters for the filter design, such as the required minimum capacity, the desired false positive rate and the ideal load factor.

In our case though, it is more important from an adversarial standpoint rather than a design need. We do not focus on the design specifics of the ICA filter, as this falls out of the scope of this project. We rather try to understand the state of the PKI ecosystem, and more specifically ICAs, in order to have a clearer view of what kind of information a potential adversary may have at its disposal.

The distribution of ICAs constitutes crucial information for the adversary. Knowing how many ICAs can be typically found in certificate chains, and how many end-entity certificates each ICA has issued affects the adversary's accuracy. For instance, imagine a certificate chain containing a single ICA. The approximate membership query performed by the adversary will succeed. Consequently the adversary will know that the client visited a domain which had that ICA in its certificate chain. If that ICA has thousands of issued end-entity certificates, the adversary has a very low chance of identifying the exact domain visited by the client among all the possible ones. If on the other hand, the ICA only has a single issued end-entity certificate, the adversary is certain that the associated domain has been visited by the client. The same is valid for chains with multiple ICAs.

4.1 Tranco List

Tranco [17] is a research-oriented ranking of top websites across the Internet. It is specifically designed to be much less variable on a day-by-day basis than most other commercial websites popularity rankings, while also implementing hardening techniques against malicious manipulation of the ranking. By default, the Tranco list is computed on a global scale, containing the most popular domains across the entirety of the web. Despite this, in our project we focus our experiments on a more restricted use case: the case of Switzerland. The decision to restrict the list to domains seen only in Switzerland was made because of various reasons.

A slightly more restricted scenario is arguably more realistic with respect to what a real malicious adversary would do in this case. An adversarial server provider would have at least some minimal information about the approximate location of possible victims. This could arise from the fact that the compromised server under its control may be offering some kind of service that operates on a local scale, or that users connecting to it can set a language to use, a shipping region or even a currency to use in case of e-commerce.

The Tranco list is particularly convenient in this scenario because it can be configured in various ways, depending on the nature and the context of the research. For example, it is possible to restrict the list to domains seen only in specific continents, regions or nations. This feature has a consequence: the domains which appear in the list are only those included in the Chrome User Experience Report [10], without including all the other lists used by Tranco. The final list used for this project has been configured this way to only include domains seen from Switzerland.

Using only website domains that are contained in the Chrome User Experience Report is a solution that only takes into account the navigation destinations of Chrome users. On one hand, this means that popular destinations among users of other browsers will not influence the list. On the other, we argue that this does not represent a problem for our scenario. On the contrary, Chrome is by far the most popular web browser, accounting for $\approx 65\%$ of market share globally, and for $\approx 61\%$ on a European level as of February 2024 [1]. For reference, the runner-up is Safari with a global market share of $\approx 18\%$, and $\approx 19\%$ in Europe. Consequently, the final domain list is more representative of the average user browsing the web, the profile that we want to investigate.

4.2 Exploratory Analysis

We performed an exploratory analysis regarding the general current state of PKI within the selected set of domains. More specifically, we collected data regarding the total amount of ICAs, and the certificate chain length distribution in order to understand typically how many ICAs can be found in each certificate chain. This information allows us to evaluate how common ICAs are in certificate chains, and whether our approach is generally applicable. If most certificate chains have no ICAs, then the scope of the privacy implications we outline is very limited. A greater presence of ICAs in certificate chains would indicate that our approach would have a broader impact.

The methodology consisted of initiating a TLS connection to each domain listed in the list under investigation, and then downloading the certificate chain sent by the server. We then recorded the total amount of distinct ICAs and how many certificates had a certain length.

We performed this analysis on three different sets of domains, containing 100000 domains each:

- Tranco list configured to include the top-ranked 100000 domains from the Chrome User Experience Report seen in Switzerland, referred to as `Tranco_CH`.
- Tranco list configured to include the top-ranked 100000 domains from the Chrome User Experience Report seen in the EU, referred to as `Tranco_EU`.
- Unconfigured Tranco list, including the top-ranked 100000 domains globally, referred to as `Tranco_World`.

The lists containing results from the Chrome User Experience Report refer to the report of November 2023. The unconfigured Tranco list with global results was created on 23 November 2023.

The results of the analysis are shown in Table 4.1. There is a noticeable difference in the amount of certificate chains that do not contain ICAs between the Swiss list and the other two: in the EU and global case there are $\approx$ 8% more chains without any ICAs. Also, certificate chains with one ICA are $\approx$ 14% more in the Swiss list compared to other two. Chains with two or more ICAs have a very similar distribution between the Swiss and global list, while the EU list has $\approx$ 6% more. Ultimately, in `Tranco_CH` certificate chains that contain at least one ICA are more than 70%, while in `Tranco_EU` and `Tranco_World` they are closer to 60%. On the other hand, the Swiss list is the one with the least amount of unique ICAs among the three, with 335 of them, significantly less compared to the other two which are closer to 400. The different distribution and number of unique ICAs between the three lists must be considered by an adversary who focuses on different possible

List Name	Total Unique ICAs	Certificate Chain Length (%)				
		0 ICAs	1 ICA	2 ICAs	3 ICAs	> 3 ICAs
Tranco_CH	335	27.82%	55.21%	16.73%	0.16%	0.08%
Tranco_EU	385	35.63%	41.77%	22.32%	0.21%	0.07%
Tranco_World	402	36.66%	46.82%	16.40%	0.09%	0.03%

Table 4.1: Total amount of distinct ICAs and their distribution among certificate chains observed in the specified domain list.

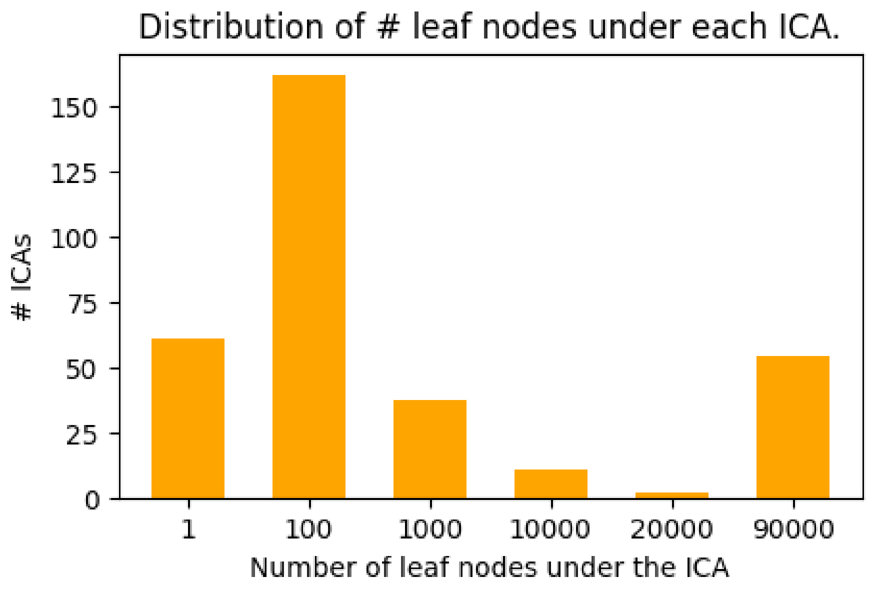

Figure 4.1: Distribution of ICAs from the Swiss list, grouped in buckets depending on the number of leaf nodes present in their subtree. The bucket indicated at each point on the x-axis includes ICAs which have a number of leaf nodes included between the previous x value (excluded) and the current one. For example, the second point on the x-axis is 100: the ICAs in this bucket have between 2 and 100 leaf nodes. The next one between 101 and 1000 and so on.

victim sets. The results show that these parameters are not uniform over the different lists. Ultimately, this could lead to a variation in the effectiveness of the attacks based on the victim's geographic location.

Finally, Figure 4.1 shows how many leaf nodes can be found in the subtree rooted in a specific ICA, i.e. how many end-entity certificates each ICA has issued. The analysis shows that ≈ 70% of the ICAs in the Swiss list have less than 100 leaf nodes. The average number of leaf nodes per ICA is ≈ 13580.

Chapter 5

Problem Modelling

Within this chapter, we present our modelling approach. With our model, we are trying to capture the scenario in which the AMQ-based ICA suppression method could be deployed on a large scale. Clients constantly update their ICA filter with each newly acquired or revoked ICA. The ICA filter contains information about all the intermediate certificates that have been encountered by the client during its web navigation. We investigate the case of a client connecting to a malicious server, to which the victim provides its ICA filter. The adversary can use the acquired ICA information to construct a list of domains that have been possibly visited by the client.

We define a model for the client who establishes a TLS connection with a server. More specifically, we outline the assumptions that we make about the client, its capabilities and limitations, which should capture the behaviour of the user we want to model. There are also honest, non-malicious servers to which the client connects prior to connecting to the malicious one. These honest servers will be modelled as part of the client model.

On the other hand, the server is modelled under the assumption that it is being controlled by a malicious adversary. Once again, the server model is composed of the assumptions that we make on the adversary, and more specifically on its capabilities and limitations. These should resemble those of a real-world attacker having access to the victim's ICA information. The adversarial goal is to leak user's sensitive data. More specifically, the adversary extracts a set of domains that the user has possibly visited information contained in the client's ICA filter.

5.1 Similarities with Website Fingerprinting

During the modelling process, it became apparent that there are multiple similarities between the scenario of a malicious server in AMQ-based ICA suppression and website fingerprinting attacks.

Website Fingerprinting (WF) [15] [26] is a traffic analysis attack, performed by a passive eavesdropper. It is used to infer the websites accessed by users, using exclusively traffic gathered from encrypted connections. This attack relies on the use of communication metadata such as packet size, inter-packet timing, direction of communication and other statistical properties in order to extract significant patterns to discern websites. All the recorded features are then usually processed by machine learning algorithms to produce distinct fingerprints of the communication traces between the clients and the corresponding visited website. These are then compared to precomputed fingerprints of known websites in the attacker's database. By analyzing similarities between the intercepted traffic and the known fingerprints, the attacker can infer which websites the user is accessing.

In our case, we assume the attacker is not a local passive eavesdropper, but is instead in control of the server to which the user will eventually connect. The adversary does not need to record encrypted traffic, as the client directly provides its ICA filter to the server. But in both cases, the adversary needs to infer websites accessed by the victim, with only minimal information about the past navigation destinations of the targeted user. We additionally assume that the adversary *only* has access to the ICA information provided by the client's filter. We restrict the adversary in such a way as to determine the impact of introducing this additional ICA information in the TLS handshake, without skewing the results with leaks obtained by other fingerprinting techniques. In a real-world scenario, the adversary may employ other techniques to further refine its attack.

In WF, there is a distinction between two different modeling approaches for the attack: *closed-world* scenario, and *open-world* scenario [15]. These are based on different assumptions about the knowledge available to the attacker regarding the set of websites being visited by the user. More specifically:

- Closed-world: the set of websites that the user may visit is very limited and is known to the adversary. This assumption is unrealistic and not reflective of real-world users' behaviour, but it is useful for comparing the performance of different classification approaches.
- Open-world: the user may visit any website, even those unknown to the adversary. The adversarial goal in this case is to identify whether the user has visited any websites belonging to a set of monitored websites. The set of websites that the adversary does not actively monitor is called the *background set*, while the set containing monitored ones is called the *foreground set*. Consequently, the final task for the adversary is to classify the recorded communication traces as background or foreground traffic.

Of course, the open-world scenario is more realistic compared to the closed-world one. On the other hand, this also entails that the attack for the adversary is much more complex. The scope is much larger: the user can practically visit any website available on the Internet. For this reason, the adversarial goal is weakened in the open-world scenario: the adversary no longer needs to exactly identify the visited website, but rather it is required to distinguish whether the user visited a known destination or not.

The scenario of our project is similar to WF because it can also be represented quite well by the open-world scenario: the user is in principle allowed to visit any website, known or unknown to the adversary. In our case, the adversary has a very different kind of information at its disposal compared to the one in WF: only ICA data is available, while in WF, even with encrypted communication, there is a lot of metadata that can be analysed to extract features with the use of machine learning models. In our case, if the adversary wants to know whether the victim has visited a certain domain, the only way to infer it is to compare the ICAs for that specific domain with the ICA information in the filter. More specifically, if the client has visited a specific domain, the ICAs in that domain's certificate chain will appear in the ICA filter. An additional level of complexity is given by the fact that the same ICA can be shared among thousands of different domains, rendering the adversarial goal even harder to achieve.

5.2 Client Model

Due to the motivations presented in the previous section, our general model is very similar to the open-world scenario of WF. Since we cannot implement a client that can actually visit *any* domain present on the Internet, we consider a *universe set U*. The universe contains all the domains included in the list generated using Tranco, as specified in Section 4.1. The list contains 100000 domains, which we deem large enough to be a sufficient approximation of the open-world scenario. The client can visit any domain within the universe set.

The client is assumed to behave honestly, i.e., it follows the AMQ-based ICA suppression protocol described by Sikeridis et al [23]. The client is also assumed to include all the ICAs that it encounters in certificate chains of the servers it connects to prior to the malicious server. None of the encountered ICAs can be excluded from the filter, nor added if they were not present in any certificate chain obtained from visited servers.

The pseudocode of the client can be found in Algorithm 1. The client establishes a connection with a certain amount of domains within the universe, receives their certificate chains and adds all the ICAs in the chain to its ICA filter. We call the set of domains visited by the client the *visited set V*. The

Algorithm 1 Pseudocode of Client

```
function Client($n, U$)
    for $i \in \{1, \ldots, n\}$ do        ▷ $n$ is the number of domains the client visits
        $d \xleftarrow{\$} U$
        $ica\_list$ ← get_ICAs($d$)
        for $ica \in ica\_list$ do
            $ICA\_Filter$.insert($ica$)
        end for
    end for
    return $ICA\_Filter, V$
end function
```

ICA filter is then sent to the server, controlled by the adversary. We do not model the entire TLS 1.3 handshake: for our purposes, we only model the part relevant to the privacy analysis and exclude the rest. The only relevant information to evaluate the privacy leakage of the AMQ-based ICA suppression is solely carried by the ICA filter.

5.3 Adversary Model

The adversary is in control of the server to which the victim connects after n visited websites. The adversary decides on a set of domains to monitor. We call this set the *monitored set* M. The following outlined approach is based on our analysis and the observed vulnerabilities observed in the AMQ-based ICA suppression method.

The adversary establishes a TLS connection with each monitored domain and obtains their respective certificate chains. The attacker then constructs a dataset which contains an entry for each domain in the monitored set, in which the ICAs of the corresponding certificate chains are stored. The pseudocode for this procedure can be found in Algorithm 2.

Algorithm 2 Pseudocode of Adversary extracting ICAs for the Monitored Set

```
function Adv_Setup($M$)
    for $d \in M$ do
        $ica\_list$ ← get_ICAs($d$)
        $domain\_ica\_map[d]$ ← $ica\_list$ ▷ Each domain is mapped to its ICAs
    end for
end function
```

Recall that the ICA filter that the server receives from the client is a Cuckoo filter, a probabilistic data structure that carries information about the dataset it contains. As such, it allows approximate membership queries, which answer to whether a certain ICA is in the filter or not, with a false positive rate ϵ.

Upon receiving the ICA filter from the client, the adversary performs an approximate membership query for all the ICAs belonging to each domain in the monitored set. If *all* the queries for ICAs of a monitored domain are successful, then the domain is included in a set which we call the *guess set* G. The pseudocode for this part of the attack can be found in Algorithm 3.

Algorithm 3 Pseudocode of Adversary

```
function ADV_ATTACK($M$, $ICA\_Filter$)
    for $d \in M$ do
        $ica\_list \leftarrow domain\_ica\_map[d]$
        $c \leftarrow$ True
        for $ica \in ica\_list$ do
            if $ICA\_Filter$.AMQ($ica$) = False then
                $c \leftarrow$ False
                break
            end if
        end for
        if $c =$ True then
            $G \leftarrow G \cup \{d\}$
        end if
    end for
    return $G$
end function
```

If the client has visited a domain that is in the monitored set, then this domain will *always* be included in the guess set G. This is a consequence of the fact that the client is assumed to behave honestly, and it will always include the ICAs obtained from a certificate chain of a visited domain in the filter. If the user has visited a domain $d \in M$ with corresponding ICAs $C_1, \ldots, C_i$, these will be included in the ICA filter sent to the adversary. The attacker's approximate membership queries on the client's ICA filter for certificates $C_1, \ldots, C_i$ will all reveal that the ICAs are known to the client. This is because the AMQ-PDS does not produce any false negatives. Consequently, that domain will be included in the guess set, i.e.,

$$d \in V \land d \in M \implies d \in G \qquad (5.1)$$

On the other hand, domains that have been inserted in the guess set *have not been* necessarily visited by the client. This is due to two main reasons.

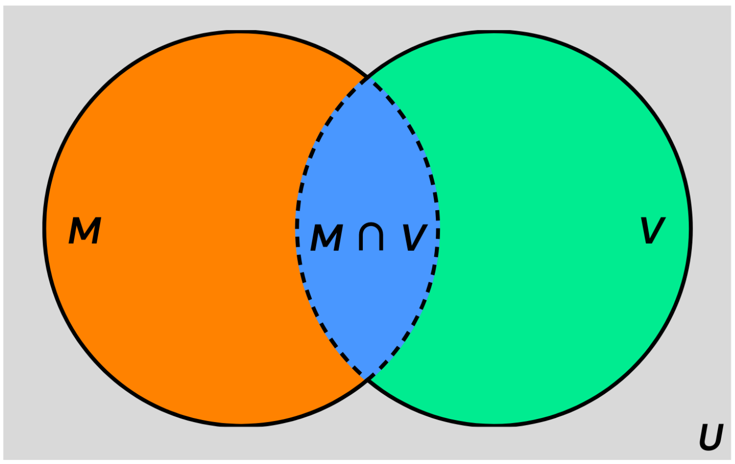

Figure 5.1: The monitored set M, the visited set V and their intersection $M \cap V$, all inside the universe set U.

The first one is that the client may have visited another domain which has the same ICAs as one (or more) domains included in the monitored set. If a domain $d \in M$ has ICAs $C_1, \dots, C_i$, there could be another domain $\tilde{d} \in V$ with ICAs $\tilde{C_1}, \dots, \tilde{C_j}$ such that $i = j$ and $C_1 = \tilde{C_1}, \dots, C_i = \tilde{C_j}$. In this case, the approximate membership queries performed by the adversary will all confirm the presence of the ICAs for domain d, which will be inserted in G. The second one is due to the probabilistic nature of the ICA filter: with probability ϵ the approximate membership query will result in a false positive. This could lead the adversary to include domains in the guess set for which the ICAs are not really known by the adversary, but the approximate membership query resulted in a false positive. Consequently, not all monitored domains in the guess set are necessarily visited by the client, i.e.

$$d \in M \wedge d \in G \nRightarrow d \in V. \tag{5.2}$$

We can better visualise the relationship between these different sets in Figure 5.1. The figure shows the monitored set M, the visited set V and their intersection $M \cap V$. Their intersection can be empty if the client does not visit any of the domains included in the monitored set.

Equation 5.1 shows that G is a superset of $M \cap V$, as it contains all domains that appear in both sets. Equation 5.2 on the other hand suggests that G may also contain domains that are not included in V. At the same time, all domains in the guess set must also be monitored domains. From these facts we can derive the following set relationships:

$$\emptyset \subseteq M \cap V \subseteq G \subseteq M. \tag{5.3}$$

5.4 Monitored Set Choice

The choice of the monitored set is arbitrary and completely in the hands of the adversary. The attacker will construct it based on what kind of information it wants to learn about the client. We can imagine that the most valuable information would be any kind of non-public, sensitive information about the user. There is a wide range of options: the user's bank or phone carrier, what e-commerce websites the user visits or even the health insurance company of the user.

In our case, we decided to build the monitored set including bank domains. In this day and age, every bank allows its customers to access an online e-banking portal from which they can carry out most of their banking needs. Knowing the exact e-banking portal that the victim uses can be considered sensitive information in the hands of malicious adversaries. An attacker could potentially gain access to the user's bank account, and steal its money through fraudulent transactions.

To construct the monitored set for our simulated adversary, we have used an official list of all authorised banks in Switzerland [8], publicly available on FINMA's website, the Swiss Financial Market Supervisory Authority. The list on the website is automatically updated every day. The list used for the experiments of this project is dated 14 January 2024. Given this list of authorised banks, we constructed another list containing all their respective web domains, which constitutes the monitored set for our adversary. The final monitored set we constructed contains 205 Swiss bank domains.

In the absence of any other additional data, we constructed the most comprehensive monitored set possible for the case of banks. This is to ensure optimal accuracy for the adversary. If there are bank domains excluded from the monitored set which could be visited by the client, this can impact the performance of the adversary in the attack. The excluded domain could potentially have the same ICAs as other bank domains in the monitored set. Consequently, these other monitored domains will be included in the adversary's guess set, without any of them having been visited by the client.

Chapter 6

Experimental Setup

Within this chapter, we present our experimental setup: the specific experiments that we designed to test our approach, as well as the metric that we used to evaluate the efficacy of the attack.

We designed two main experiments, which both rely on the client sending its ICA filter to the server, constructed as described in Section 5.2. Upon receiving the filter, the adversary constructs its guess set as described in Section 5.3. The two experiments differ in the assumptions that we make on the client's behaviour, and how the adversary adapts its attack to these different assumptions.

In our experiments, we never actually establish a TLS 1.3 connection between the client and the server. The client simply passes the ICA filter alone to the server. We purposefully omit the rest of the TLS connection, as for our purposes the required information is contained in the ICA filter itself. Implementing an actual TLS connection would only introduce further communication overhead and significantly slow down the experimentation process, without bringing any additional value for our analysis. The performance analysis of the proposed ICA suppression method is out of the scope of this project, and it can already be found in the original paper [23].

6.1 Single Connection

In this experiment, we assume a scenario in which the client performs a single connection with the malicious server. After visiting a certain number of domains and inserting the collected ICAs in the filter, it connects to the server under the control of the adversary and sends over its ICA filter.

Given our chosen monitored set of bank domains, it is reasonable to assume that the client residing in Switzerland, during its navigation, will visit at least one of them. We expect the client to, at the very least, connect to the domain of its e-banking portal. It may also visit other bank domains, e.g., to compare interest rates or because the client owns multiple bank accounts. The different possible navigation patterns of a client will be modelled by varying the number of domains visited by the client in the experiments. Some clients may make limited use of the Internet and only visit very few domains, while others may visit many more. In either case, it is safe to assume that we expect normal Internet users to access their e-banking portals at some point during their navigation activity. For this reason, we explicitly force the client to connect to a monitored domain in its first connection. Equivalently, we model the client to visit at least one domain in the monitored set:

$$\emptyset \subsetneq M \cap V \iff |M \cap V| \geq 1. \tag{6.1}$$

We have designed two different instances of this experiment:

- **Option 1 - Single Monitored Domain (SMD)**: the client visits *exactly* one domain from the monitored set M, and all the other visited domains are selected from $U \setminus M$. Consequently:

$$|M \cap V| = 1. \tag{6.2}$$

- **Option 2 - Multiple Monitored Domains (MMD)**: the client visits *at least* one domain from the monitored set M: first, it visits one domain from M; the rest of the visited domains are selected from $U \setminus M$ with probability $(1 - \rho)$, or from M with probability ρ. Consequently:

$$|M \cap V| \geq 1. \tag{6.3}$$

All the domains selected from a set are randomly sampled unless specified otherwise in the specific experiment.

6.1.1 Option 1 - Single Monitored Domain (SMD)

In order to implement this experiment, we slightly modify the client's pseudocode presented in Section 5.2. The new pseudocode is shown in Algorithm 4. We force the client to connect to a domain sampled from the monitored set in its first connection. All the remaining domains are randomly sampled from $U \setminus M$, to ensure the client does not connect to any other monitored domain apart from the first one. On the other hand, the adversary behaves exactly as shown in Section 5.3.

6.1.2 Option 2 - Multiple Monitored Domains (MMD)

Also in this instance, the client is slightly modified, as shown in Algorithm 5. We still make sure that the client first connects to a domain sampled

Algorithm 4 Pseudocode of Client - Single connection: SMD

```
function CLIENT($n, U, M$)
    for $i \in \{1, \dots, n\}$ do        ▷ $n$ is the number of domains the client visits
        if $i = 1$ then
            $d \leftarrow\$ M$
        else
            $d \leftarrow\$ U \setminus M$
        end if
        $conn \leftarrow$ TLS_connect($d$)
        $chain \leftarrow$ get_certificate_chain($conn$)
        $ica\_list \leftarrow chain[1 : -1]$
        for $ica \in ica\_list$ do
            $ICA\_Filter$.insert($ica$)
        end for
    end for
    return $ICA\_Filter$
end function
```

from the monitored set. For all the remaining domains, we condition the choice of the set to sample on the probability ρ. This practically means that each domain will be randomly sampled from M with probability ρ, or randomly sampled from $U \setminus M$ with probability $(1 - \rho)$. For example, if $\rho = 0.1$, i.e., a 10% probability, then the expected number of visited monitored domains over a total of 100 visited domains is 10. In our experiments, we use values of $\rho = 0.005, 0.01, 0.02$, which respectively translate to an additional expected amount of visited monitored domains of $0.5, 1, 2$ over a total of 100 visited domains. We consider these numbers realistic with respect to the kind of user we want to represent. Also, the different values of ρ enable us to represent even users who may have bank accounts with a number of different banks, and thus consistently visit multiple monitored domains.

6.2 Repeated Connection

In this experiment, we additionally assume that the server is able to link separate TLS connections to the same user. More specifically, we assume that between each client connection, its ICA filter is reset. The client connects to the same malicious server multiple times, or different servers monitored by the same adversary. We focus on the prior, but the approach naturally extends to the latter. We additionally assume that the client always visits the same monitored domain between different connections to the adversary. It is reasonable to consider such a scenario because, for example, most users probably have one bank account that is accessed through an e-banking por-

Algorithm 5 Pseudocode of Client - Single connection: MMD

```
Require: $0 \leq \rho \leq 1$
  function Client($n, U, M, \rho$)
      for $i \in \{1, \dots, n\}$ do        ▷ $n$ is the number of domains the client visits
          if $i = 1$ then
              $d \leftarrow_\$ M$
          else
              if $rand() < \rho$ then        ▷ $0 \leq rand() \leq 1$ randomly sampled
                  $d \leftarrow_\$ M$
              else
                  $d \leftarrow_\$ U \setminus M$
              end if
          end if
          $conn \leftarrow$ TLS_connect($d$)
          $chain \leftarrow$ get_certificate_chain($conn$)
          $ica\_list \leftarrow chain[1 : -1]$
          for $ica \in ica\_list$ do
              $ICA\_Filter$.insert($ica$)
          end for
      end for
      return $ICA\_Filter$
  end function
```

tal, which is regularly accessed. The adversary can leverage this fact, and try to reduce the size of the guess set to improve the accuracy of the attack. Since there is one monitored domain which is always visited by the client, the ICAs for that domain will always be present in all the ICA filters received by the attacker. The adversary can take the intersection of all the distinct guess sets constructed individually at each connection and use this intersection set as its final guess set. By doing so, the adversary only takes into account the domains for which all corresponding ICAs were in the ICA filter for *every* individual connection. We do not make any other specific assumptions about the rest of the domains visited by the client: they are randomly sampled as described either in SMD or MMD.

The experiment can be run either with a client that behaves as in SMD shown in Algorithm 4, or as in MMD shown in Algorithm 5. In both cases, the pseudocode for the client will be the same as the one presented for the chosen option, except for the first domain to visit. This domain will not be randomly sampled from M at each connection, but rather is randomly chosen once and stays the same in the following connections. The adversary for this experiment uses the general attack described in Algorithm 3 in Section 5.3 as a sub-routine. The pseudocode of the adversary against the repeated connection scenario is shown in Algorithm 6.

Algorithm 6 Experiment Definition - Repeated Connection

```
Require: $m \geq 0$                                ▷ $m$ is the amount of repeated connections
  function ADV_ATTACK_REPEATED($M$)
      for $i \in \{1, \ldots, m\}$ do
          $ICA\_Filter \leftarrow$ Client($n, U, M, \rho^*$)          ▷ $\rho$ is optional
          $\tilde{G} \leftarrow Adv\_Attack(M, ICA\_Filter)$
          $G \leftarrow G \cap \tilde{G}$
      end for
      return $G$
  end function
```

6.3 Evaluation Metric - Jaccard Index

We want to define a precise quantitative metric to evaluate the performance of the adversary in each experiment. The adversary has no control over the domains from the universe which will be included in the visited set V of the client. The attacker can define and arbitrarily select the domains to include in the monitored set M. More specifically, the adversary will strategically select the set of monitored domains based on the kind of information the attacker wants to extract from the ICA filter. In our case, we chose bank domains as they may be particularly interesting for general malicious entities, but this could vary depending on the attacker's intent. For example, another plausible category for monitored domains could be mobile carriers, e-commerce websites or university login pages.

The adversarial goal in this case is to extract information about the client's visited domains from the information leaked by the provided ICA filter. More specifically, the adversary's goal is to learn the set $M \cap V$, i.e., the monitored domains that the client has visited. Intuitively, the best-case scenario for the adversary would be to construct a guess set $G = M \cap V$. As stated in Equation 5.1 in Section 5.3, all the domains in $M \cap V$ will naturally all be included in G, as there are no false negatives. At the same time, the guess set can be no larger than M as all the domains included in the guess set are also monitored domains.

Our chosen performance metric must reflect these properties of the defined sets. The client can do no better than exactly guess a set $G = M \cap V$, while the worst-case guess would be a set $G = M$. We consider this to be the worst-case scenario as it reflects the case in which the information contained in the ICA filter is too general. It does not enable the adversary to restrict its guess set compared to the initial full monitored set.

For these reasons, we have selected as our final evaluation metric the **Jaccard index**, or **Jaccard similarity coefficient**. It is a statistic used to compare two sample sets. The similarity index of sets A and B is computed as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, 0 \leq J(A, B) \leq 1. \tag{6.4}$$

It ranges from 0 to 1, where 1 indicates that the two sets are identical, i.e., they have all elements in common, and 0 indicates no similarity between the sets, i.e., they have no elements in common.

In our case, we use the Jaccard index in order to evaluate the similarity between the guess set G and $M \cap V$, i.e., we compute $J(G, M \cap V)$. A Jaccard index of 1 would indicate that the adversary has perfectly guessed all the monitored domains visited by the client. A Jaccard index smaller than 1 would indicate that the adversary has wrongfully added monitored domains that were not visited by the client in the guess set. Finally, a Jaccard index of 0 would only occur if $|M \cap V| = 0$, which would indicate that the client has not visited a single monitored domain. This instance never occurs within our experiments as the client is always forced to visit at least one monitored domain.

The Jaccard index reflects all the properties we desire from a performance metric, and consequently, it is the one we used in order to evaluate adversarial performance in our experiments in the following chapter.

Chapter 7

Results

Within this chapter, we show the data collected from the single connection experiment. We first consider the full monitored set of Swiss banks, containing 205 domains (see Section 5.4), and then use a reduced monitored set, containing just 20 randomly sampled monitored domains. Then, we present the data collected from the repeated connection experiment. We again use the full monitored set first, and then a reduced one. More specifically, we show the performance of the adversary represented by the average computed Jaccard index between the sets G and $M \cap V$. We additionally provide data about the average size of the guess set constructed by the adversary. We also show how the performance of the adversary is related to the change in universe size. Finally, we present a possible addition to the original ICA suppression method by Sikeridis et al. [23]. The proposed method strictly provides the suppression of intermediate certificates, while we show that also root CAs could be potentially suppressed, reducing the communication overhead even more, causing no significant privacy repercussions.

For every experiment, we consider 3 variants. In each variant, the client visits and collects ICAs from 100, 200 and 400 different domains, respectively. We chose these numbers in order to represent three different possible user profiles: from a not particularly active user who only visits 100 domains, up to a very active user who visits 400 domains. The results displayed in this chapter are obtained by averaging over 500 experiment runs.

7.1 Single Connection

In this experiment, the client visits 100, 200 or 400 domains, and inserts the collected ICAs in its ICA filter later on. The adversary uses the filter in order to construct its guess set, as described in Algorithm 3 in Section 5.3. We compute the Jaccard index of the guess set G and the visited monitored

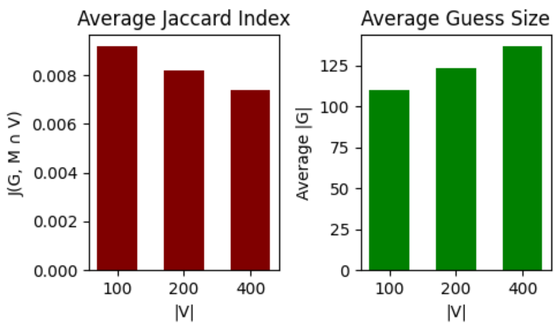

Figure 7.1: Single connection - SMD. The x values indicate the number of domains visited by the client.

domains by the client $M \cap V$. We additionally show the average size of the guess set G.

7.1.1 Single Connection - Single Monitored Domain (SMD)

This variant of the single connection experiment is characterised by the fact that the client visits *exactly one* monitored domain. Note that, as explained in Section 6.1, this entails that $|M \cap V| = 1$.

The results are shown in Figure 7.1. Under the assumptions of the single connection SMD, the adversary performs very poorly, with an average Jaccard index that oscillates between 0.009 and 0.007 in the three instances, as can be seen in the left chart of Figure 7.1. Since the size of the set containing the monitored domains visited by the client, i.e., $|M \cap V|$, is constant and is always 1, the Jaccard index only depends on the size of the guess set $|G|$. More specifically, in SMD:

$$J(G, M \cap V) = \frac{1}{|G|}. \tag{7.1}$$

They are inversely proportional, and this relation is very clear in the two charts: a larger guess size results in a smaller Jaccard index. The guess size naturally increases together with the number of total domains visited by the client. As the user collects more ICAs, there is a higher probability for each monitored domain to have matching ICAs with the ones present in the filter, and consequently be included in the guess set.

While the client only ever visits a single monitored domain, the adversary has 205 domains in its monitored set. The resulting guess set constructed by the adversary contains over 100 domains in all three instances, arriving at almost 150 when the client visits 400 domains. In the end, the adversary has restricted its original monitored set by almost 50% in the best instance. Despite that, the resulting guess set is still too large to provide any kind of

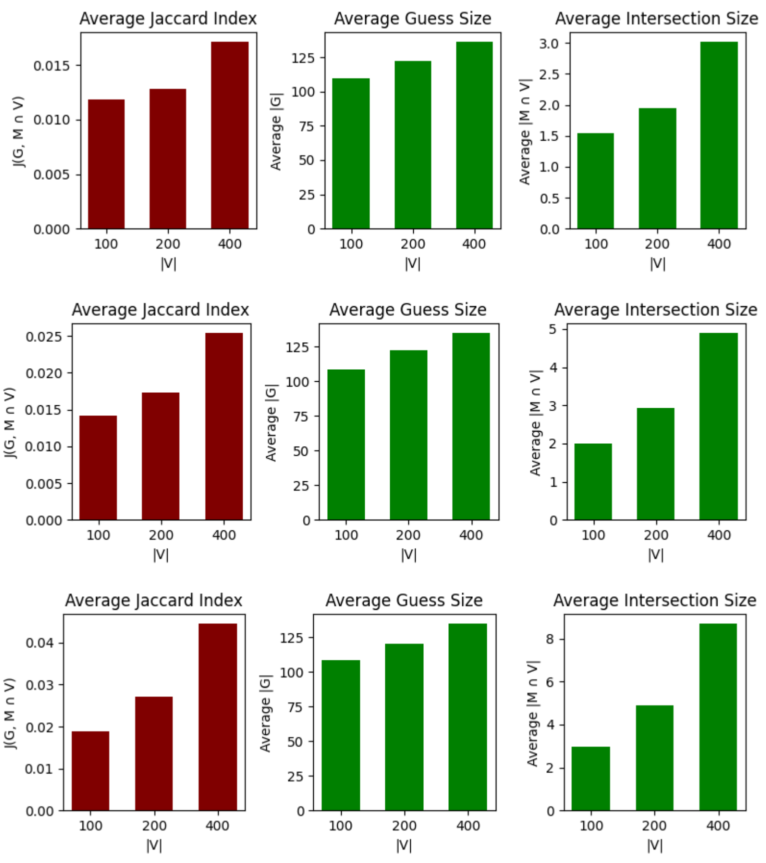

Figure 7.2: Single connection - MMD: $\rho = 0.005$ (top), $\rho = 0.01$ (middle), $\rho = 0.02$ (bottom).

valuable information to the attacker. This is due to the large ICA overlap between multiple domains in the monitored set. When multiple monitored domains share the same ICAs, they will either be all inserted together in the guess set or will all be excluded from it. This reduces the granularity of the final guess output by the adversary, which cannot include single domains in its guess, but rather groups with multiple domains.

7.1.2 Single Connection - Multiple Monitored Domains (MMD)

In this variant of the single connection experiment, the client is guaranteed to visit *at least one* monitored domain. We force the first visited domain by the client to be a monitored domain. For all the following visited domains, there is a small probability ρ that they are randomly sampled from M instead of $U \setminus M$. We have tested this experiment with values of $\rho \in \{0.005, 0.01, 0.02\}$. With these values, the expected amount of monitored domains visited by the client is respectively increased by $0.5, 1$ and 2 for every 100 total visited domains.

The results are shown in Figure 7.2. In this variant, contrary to SMD, the client can potentially visit a variable amount of monitored domains. As explained in Section 6.1, the size of $M \cap V$ is not fixed, but rather $|M \cap V| \geq 1$. Thus, we also present results on the average size of the intersection between the monitored set and the visited set, i.e., $|M \cap V|$.

The average Jaccard index values are still very low, but higher than those seen in SMD. More specifically, they are always below 0.05, even in the best-case outcome for the adversary. The Jaccard index grows with the value of ρ: as the latter doubles, the former increases by more than ≈ 1.5 times. We see that, as the client visits more monitored domains in the same total visited domains, the Jaccard index increases.

Another notable aspect transpires from the obtained results: the average guess size remains practically equal for all the tested ρ values, and also the same as the one recorded in the SMD experiments. This suggests that the size of the constructed guess set, assuming that the monitored set is fixed, depends solely on the size of the visited set $|V|$. This result is consistent with how the guess set is constructed. In our model, the only information available to the adversary is the client's ICA filter. The filter will naturally contain more distinct ICAs as the client visits more distinct domains, and the adversary will find more monitored domains which can be included in the guess set. This process does not directly depend on the number of monitored domains that the client visits. Rather, it depends on the number of ICAs collected and how many monitored domains' ICAs are present in the filter.

While the average guess size remains constant as the values of ρ increase, the average intersection size becomes larger, since the client visits more monitored domains. This results in a larger Jaccard index value: the guess set is the same as in SMD, but in this variant, there are multiple domains in G which have been visited by the client. Ultimately, the guess set obtained in this variant is slightly more informative for the adversary compared to SMD.

7.1.3 Reduced Monitored Set

The results obtained from the previous experiments show poor effectiveness of the attack. One of the reasons for it may be the fact that the monitored set used is too large to extract any valuable information from the ICA filter. In some cases, the adversary might have some additional information about the client. For example, more precise location information might lead to the exclusion of many domains from the monitored set. A user located in a certain area is much more likely to take advantage of banks that operate there. For instance, someone living in Lugano is unlikely to have a bank account with the Cantonal Bank of Geneva.

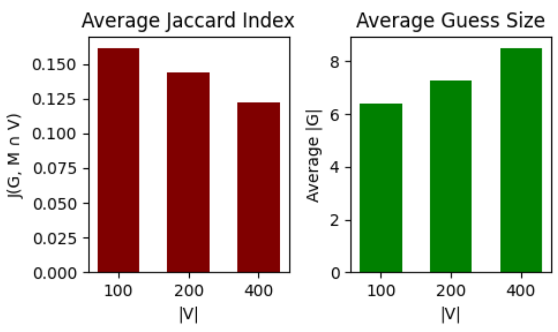

Figure 7.3: Single connection - SMD: $|M| = 20$.

For this reason, we also experimented with much smaller monitored sets. More specifically, we randomly chose 20 bank domains from the original full set of 205 and used them as the new adversary's monitored set. We ran the same experiments with this new monitored set and collected the respective results.

Figure 7.3 shows the results obtained with SMD. As expected, the trends of the average Jaccard index and the average guess size follow those from Figure 7.1: the guess size still increases with the number of domains visited by the client, while the similarity index decreases. On the other hand, the value of the Jaccard index is more than 15 times larger than the one obtained with the full bank list, while having a monitored set that is more than$\approx$ 10 times smaller.

Ultimately, the guess size contains on average between 6 and 9 domains, depending on the size of the visited set. The results obtained in this instance are more promising: the constructed guess size is $\approx$ 70% smaller compared to the initial monitored set, ultimately containing only a few domains. This means that the adversary can restrict the client's navigation destinations in a much more effective manner by having a significantly smaller monitored set.

Figure 7.4 shows the results obtained for MMD with the reduced monitored set. The trends for the recorded values are again the same as with the full monitored set. Here the Jaccard index is generally $\approx$ 10 times larger than when using the full banks list. In the variant with $|V| = 400$ and $\rho = 0.02$, the Jaccard index reaches a value of $\approx$ 0.4. In this instance, the adversary constructs a guess set in which more than 1 in 3 included domains have been actually visited by the client.

These results show that with a smaller monitored set, this kind of attack can become practically viable for real-world adversaries. Additionally, the monitored set used for this experiment is composed of randomly chosen

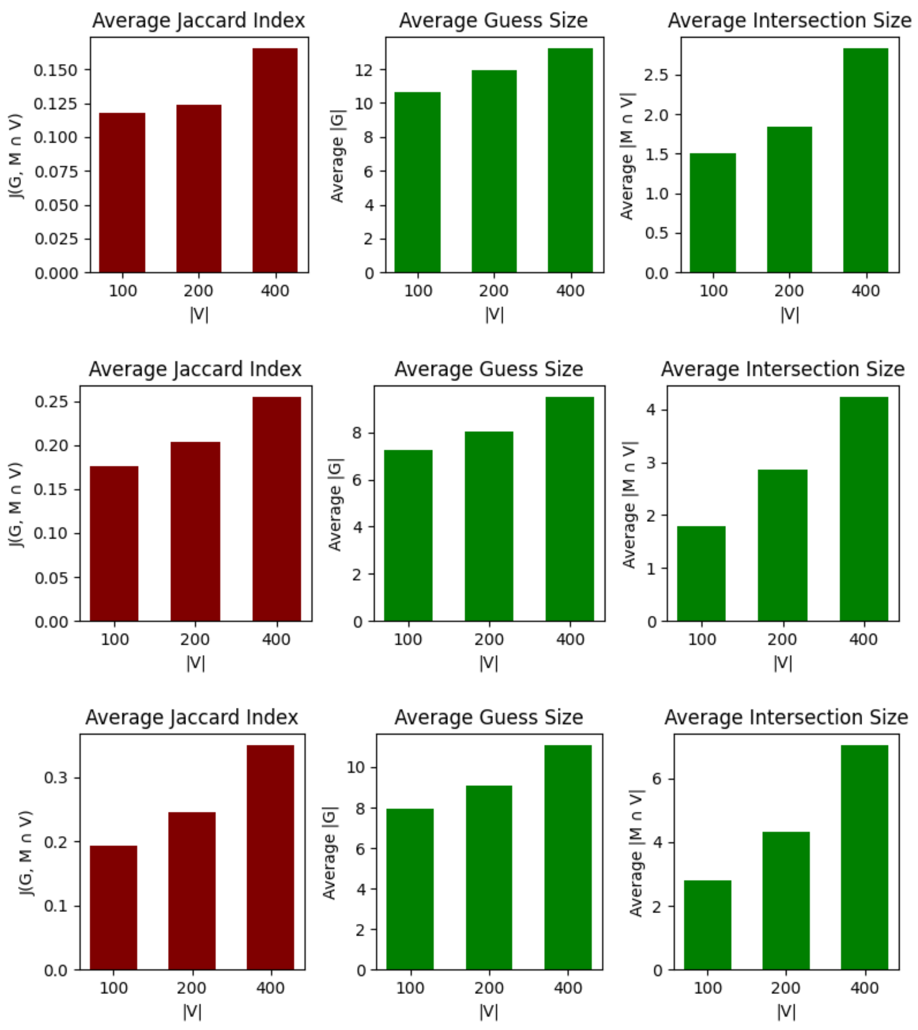

Figure 7.4: Single connection - MMD: $\rho = 0.005$ (top), $\rho = 0.01$ (middle), $\rho = 0.02$ (bottom).

monitored domains. With a more in-depth and targeted analysis of the victim, it may be possible to optimize the choice of the domains to include in the monitored set and further improve the effectiveness of the attack.

7.1.4 Correlation with Universe Size

We collected additional data about the correlation between the effectiveness of the attack and the size of the universe set.

In all of our experiments, we used the largest universe at our disposal, containing 100000 domains, as presented in section 4.2. We generally deem this large enough to be representative of the size of the navigation universe of a general Internet user. Despite this, it is important to understand whether

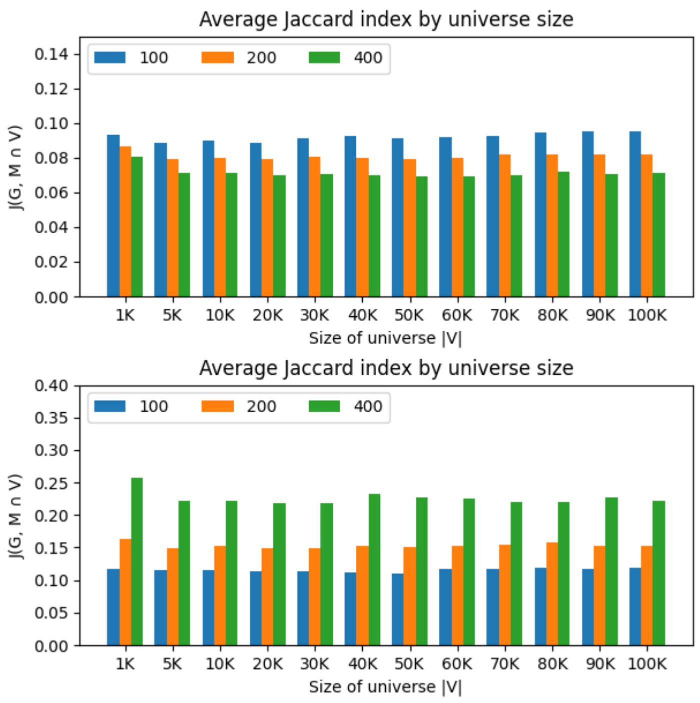

Figure 7.5: Correlation between Jaccard index value and universe size ($|M| = 20$): SMD (top) and MMD (bottom).

there is a relation between the size of the universe set and the performance of an adversary. This kind of relation could lead to different ways to optimise the attack, depending on the kind of user that is being targeted.

In order to collect the necessary data, we ran both SMD and MMD experiments with various universe sizes and recorded the average Jaccard index resulting from each size. The experiments were performed using the reduced monitored set.

The collected data, displayed in Figure 7.5, shows no significant correlation between the performance of the adversary and the universe size. The recorded Jaccard index values show no noticeable variation, apart from the expected small fluctuations due to the randomness present in the experiments.

7.2 Repeated Connection

In a real-world scenario, a client may realistically visit the same server multiple times. An adversary controlling that server may gain an advantage from collecting the client's ICA information at different points in time. This is the kind of occurrence that we model with this experiment.

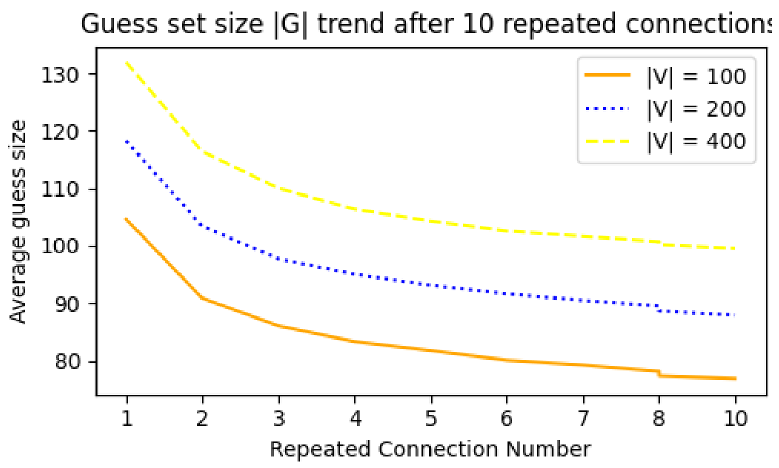

Figure 7.6: Repeated connection - SMD: average guess set size after each repeated connection.

In this experiment, the client connects to the same malicious server multiple times. After each connection, the ICA filter is reset. As in the single connection case, the client visits and inserts into its filter ICAs from 100, 200 and 400 domains. We assume that between each different connection to the malicious server, the client always visits the same monitored domain. If the client runs as in SMD, that monitored domain is also the only one that it visits. If it runs as in MMD, it may additionally visit other ones, following the usual mechanism for this variant. The adversary uses the filters to construct one guess set per each client's connection. Finally, the adversary's final guess is the intersection of its individual guess sets. The adversarial goal is to infer the one monitored domain that is always visited by the client.

In the experiments with repeated connection, the client connects to the malicious server 10 times. We have set the number of repeated connections to 10. We have experimentally seen that after the first couple of connections, the size of the constructed final guess set plateaus and does not decrease any more. Thus, 10 repeated connections are more than enough to show the effects of this attack.

For each experiment, we show the average guess set size for each repeated connection obtained from the intersection of all previous guesses.

7.2.1 SMD

In this case, the client always visits exactly one monitored domain, which is always the same throughout all the different connections.

Figure 7.6 shows the results collected in this experiment. There is an average decrease in the guess set size of $\approx 25\%$ for all three visited set sizes. This is a significant improvement over the single connection case, but ultimately

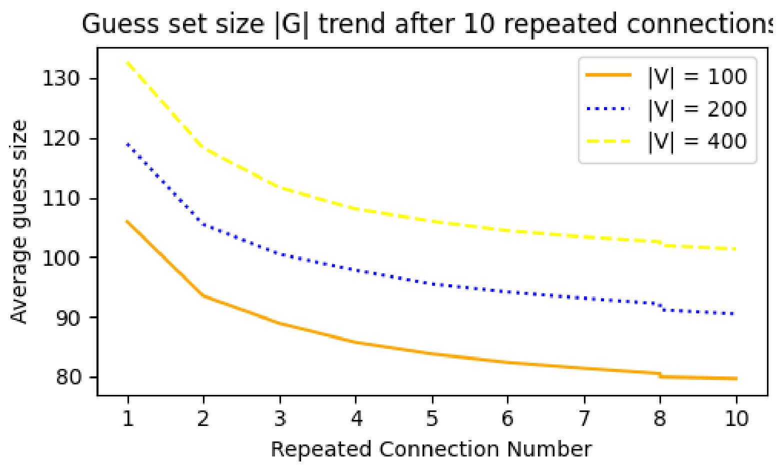

Figure 7.7: Repeated connection - MMD ($\rho = 0.01$): average guess set size after each repeated connection.

the final guess size is still too large in order to consider this an effective privacy attack. The final guess size is between 80 and 100 domains in all three cases, while the adversary should guess one single monitored domain that has been visited by the client. This is due to the monitored set being too large.

7.2.2 MMD

In this instance, between each connection, the client is allowed to visit other monitored domains in addition to the one that is always visited in each connection.

Figure 7.7 only shows the results for $\rho = 0.01$, as the ones collected for the other ρ values are comparable and do not significantly differ from the ones displayed in the figure. Also in this case, the repeated connections prove to be beneficial for the adversary, with an average decrease in guess set size of $\approx 20\%$. Nonetheless, in this instance, it remains true that the size of the monitored set is too large to allow the adversary to effectively attack the victim's privacy.

7.2.3 Reduced Monitored Set

We reduce the monitored set to 20 domains for the repeated connection scenario as well. The 20 monitored domains are once again randomly chosen from the full bank domains list.

Figure 7.8 shows the results obtained for SMD with the reduced monitored set. We observe that the initial guess set size naturally increases with the visited set size, as it does in the single connection case. In this case, the

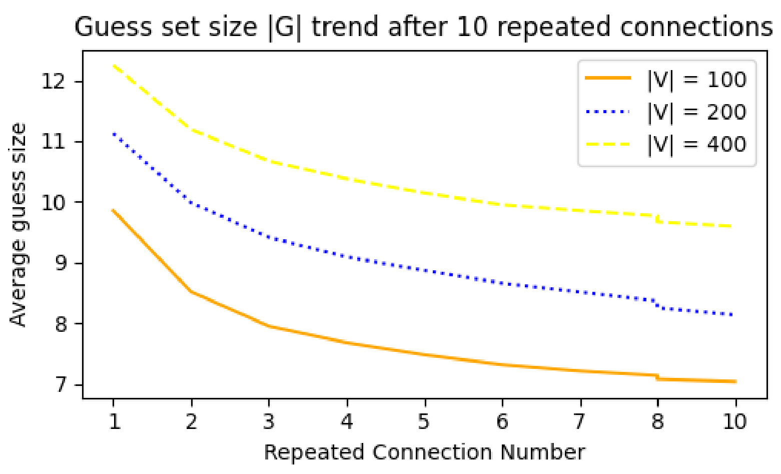

Figure 7.8: Repeated connection - SMD ($|M| = 20$): average guess set size after each repeated connection.

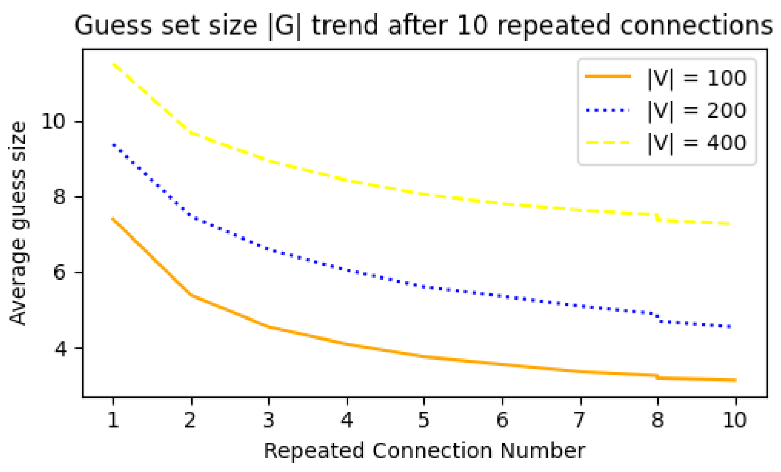

Figure 7.9: Repeated connection - MMD ($\rho = 0.01, |M| = 20$): average guess set size after each repeated connection.

guess set size is reduced on average by $\approx 30\%$ in the $|V| = 100$ variant, a significant improvement over the single connection experiment. Also, using a smaller monitored set ensures that the number of domains included in the final guess set is relatively small. Ultimately, the adversary can restrict its initial monitored set of 20 domains to a final guess set of around 7 domains. This kind of guess set size is much more useful for the adversary, as it narrows down the possible visited monitored domains to a very limited set compared to the single connection experiment.

Figure 7.9 shows the results obtained for MMD with the reduced monitored set. Only the results for $\rho = 0.01$ are displayed, since the ones obtained with the other values do not significantly differ from the results in the figure. In this case, the decrease in the guess set size is even more consistent compared to those for SMD. For $|V| = 100$ and $|V| = 200$ the initial guess set size is reduced by almost 50%.

Ultimately, the results obtained in the repeated connection experiment with the reduced monitored set are very promising for the attacker: the constructed guess sets are very limited, containing only a few domains, and the required amount of repeated connections is quite low and reasonable for a real-world scenario. In addition, the adversary does not necessarily need to wait for exactly 10 repeated connections, but can gradually reduce the guess set via the intersection method after each client connection.

This result highlights the importance for the adversary to have access to additional information about the client. We have purposefully limited and restricted our designed adversary when considering the full monitored set, in order to only account for the privacy leakage of the AMQ-based ICA suppression. Nonetheless, the reduced monitored set is more representative of a real adversary, which can gather information about the victim through all the methods at its disposal.

7.3 Alternative Domain Lists

All the results presented in this chapter are collected using the Swiss domain list presented in Section 4.2 as the universe set. This is because we were able to obtain official data on all the authorised banks in Switzerland. We did not do the same for the EU and global domain lists. Part of the reason is that considering all the authorised banks in the EU, or even the world, would result in monitored sets too large to be useful for the adversary: there would be too many domains sharing the same ICAs, and the final guess would not be as informative for the attacker compared to a much smaller guess set. Also, on a practical level, gathering the same information about the authorised banks in such a large geographical area is much more complex and time-consuming compared to a single country. For this reason, time constraints did not allow us to design an alternative approach and to perform a deeper analysis of possible monitored sets to use for the EU and global cases.

Nonetheless, we have collected data also for the EU and global Tranco lists. The monitored set used by the adversary was composed of bank domains chosen arbitrarily among the most popular banks in the EU and the world. We prefer to focus this project on the Swiss use case, which we were able to analyse more in-depth and in a more rigorous way. In addition, the results obtained with the EU and global lists are extremely similar to the ones displayed in this section using the Swiss list. We have not noticed any significant privacy consequences in considering one list over the other as a universe set. This shows that our approach is not specific to Switzerland, but it can be generalised to other settings as well.

7.4 Root CA Suppression

The TLS 1.3 specification [19] allows for the root CA certificate to be omitted from the handshake under the assumption that the other entity already possesses it in order to validate its peers. The proposed AMQ-based ICA suppression method by Sikeridis et al. [23] only specifies the suppression of intermediate certificates, without ever referring to root CA certificates.

Suppressing root CAs in addition to ICAs, would further reduce the communication overhead introduced by PQ signatures in TLS 1.3. For this reason, we propose the addition of root CAs in the client filter to the original ICA suppression method.

We have adapted the client and adversary model used in our experiments to the root CA suppression. We have tested the modified method by performing all the same experiments presented in this chapter. Ultimately, the results for the attacker are identical to the case of the originally proposed ICA suppression method. Thus, in our opinion, suppressing root CAs does not result in any additional privacy leakage compared to the case in which exclusively ICAs are suppressed.

Chapter 8

Conclusion

In this project, we tackle the problem of evaluating the privacy implications of AMQ-based PQ TLS authentication. In particular, we evaluate the privacy leakage introduced by the AMQ-based ICA suppression method proposed by Sikeridis et al. [23].

Our contributions start with an exploratory analysis of the current state of PKI, presented in Section 4.2. More specifically, we study how ICAs are distributed in the wild: how many unique ICAs are there, how many can we expect to find in each certificate chain and how many end entity certificates have they issued on average. We discuss the privacy leakage introduced by the proposed ICA suppression method in Section 3.2 and we define a measure for the leakage in Section 6.3. We design both a client and an adversarial model. We also design various attack protocols for the adversary under different assumptions. We outline the experimental setup used to evaluate the effectiveness of the designed attacks in Chapter 6. We then present the collected results in Chapter 7. We finally propose the addition of root CA suppression to the original ICA suppression method in Section 7.4, also showing that this enhancement would not have any additional privacy consequences.

The strength of our model lies in the fact that the adversary has complete control over the choice of the monitored set. As such, the adversary has a great amount of flexibility, making it hard for the client to put in place effective mitigation techniques.

During our research we tried to design some mitigation solutions: we have considered adding a fixed small amount of additional ICAs in the ICA filter even if the client did not visit any domains with those intermediate certificates. This addition would cause the adversary to wrongly insert additional monitored domains in its guess set, decreasing the accuracy and effectiveness of the attack. By adding only a fixed limited amount of ICAs, we do not burden the client with the additional storage of too many PQ certificate signatures. The problem we run into when designing this mitigation solution is that its effectiveness strongly depends on the specific monitored set chosen by the adversary, over which the client has no control or knowledge. If the additional ICAs inserted in the filter do not match any of the monitored domains' ICAs, the final guess set constructed by the adversary will be unaffected.

Ultimately, after our evaluation, the AMQ-based ICA suppression method proposed by Sikeridis et al. [23] does not seem constitute a big privacy risk for users by itself. On the other hand, it is a mechanism that can potentially be deployed on a large-scale complex system such as the TLS infrastructure. Because of that, it cannot be analysed solely on its own, in a vacuum, but there needs to be a careful analysis of its interactions with other components of the system. For instance, our results show that an adversary that is limited to only the client's ICA data performs poorly. On the other hand, when the adversary is given more power, under more realistic assumptions, it is able to collect additional information about the client. This additional information directly translates into a greatly improved adversary effectiveness, with results that could be worrisome in a real-world setting.

8.1 Future Work

We mainly designed attacks in which the adversary only makes use of the client's ICA information and no other additional information. The obtained results are satisfactory, but in a real-world scenario, the adversary does not have any limitations. The attacks presented in this project could be used in parallel with some other passive network attacks on the victim, such as state-of-the-art website fingerprinting techniques [15] [26] [5]. For instance, website fingerprinting could potentially help the adversary in reducing the size of the monitored set, which is crucial for the efficiency of the proposed attacks.

The most important aspect for the adversary, and the one that ultimately defines which information is going to be extracted from the ICA filter, is the choice of the monitored set. Given the time constraints of this project, it was not possible to perform an in-depth analysis of how to optimize the monitored set in order to maximize the adversary's effectiveness. For example, having multiple domains which share the same ICAs in the monitored set is detrimental for the adversary, as these will always all appear in the guess set together, even if the client only visited one of them. Future research on this topic could focus on ways to enable the adversary to have monitored domains which do not share ICAs.

We have only tested our experiments on monitored sets containing bank domains, but in a real-world scenario, the adversary has complete control over the choice of the monitored set. Future research could study how the type of monitored set impact the effectiveness of the attacks.

In this work, we have mainly focused on the adversarial setting, but there is also room to improve the privacy leakage of the ICA suppression method. Designing mitigation solutions could, for example, require the client to include additional ICAs in its filter to reduce the adversary's accuracy. This would require the client to locally store additional PQ signatures resulting in a potentially high storage cost. There is an interesting tradeoff between privacy and utility that needs to be analysed in-depth to implement some effective privacy mitigations. In addition, the client has no control over the choice of the monitored set by the adversary. This makes the design of mitigation solutions even more complex, as they need to protect against a very broad set of possible monitored sets.

Bibliography

[1] Global market share held by leading internet browsers from January 2012 to February 2024 [Graph], StatCounter, February 7, 2024. [Online]. Available: https://www.statista.com/statistics/268254/market-share-of-internet-browsers-worldwide-since-2009/.

[2] Google Transparency Report, 2024. https://transparencyreport.google.com/https/overview.

[3] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[4] Chia-ling Chan, Romain Fontugne, Kenjiro Cho, and Shigeki Goto. Monitoring tls adoption using backbone and edge traffic. pages 208–213, 04 2018.

[5] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 753–770, Boston, MA, August 2022. USENIX Association.

[6] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.

[7] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, page 75–88, New York, NY, USA, 2014. Association for Computing Machinery.

[8] FINMA. Authorised Banks and Securities Firms. `https://www.finma.ch/en/~/media/finma/dokumente/bewilligungstraeger/pdf/beh.pdf?la=de`.

[9] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. 2019.

[10] Google. Chrome User Experience Report. `https://developer.chrome.com/docs/crux`.

[11] Dennis Jackson. Abridged Compression for WebPKI Certificates. Internet-Draft draft-ietf-tls-cert-abridge-00, Internet Engineering Task Force, September 2023. Work in Progress.

[12] Panos Kampanakis and Michael Kallitsis. Faster post-quantum tls handshakes without intermediate ca certificates. In Shlomi Dolev, Jonathan Katz, and Amnon Meisels, editors, *Cyber Security, Cryptology, and Machine Learning*, pages 337–355, Cham, 2022. Springer International Publishing.

[13] Myounghoon Kim, Joon Suh, and Hunyeong Kwon. A study of the emerging trends in sim swapping crime and effective countermeasures. In *2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD)*, pages 240–245, 2022.

[14] NIST. Post quantum cryptography. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.

[15] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. Website fingerprinting at internet scale. 02 2016.

[16] Prashant Pandey, Michael A. Bender, Rob Johnson, and Rob Patro. A general-purpose counting filter: Making every bit count. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 775–787, New York, NY, USA, 2017. Association for Computing Machinery.

[17] Victor Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. 01 2019.

[18] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2004.

[19] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[20] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-17, Internet Engineering Task Force, October 2023. Work in Progress.

[21] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. Cryptology ePrint Archive, Paper 2020/534, 2020. https://eprint.iacr.org/2020/534.

[22] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[23] Dimitrios Sikeridis, Sean Huntley, David Ott, and Michael Devetsikiotis. Intermediate certificate suppression in post-quantum tls: an approximate membership querying approach. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '22, page 35–42, New York, NY, USA, 2022. Association for Computing Machinery.

[24] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. Post-quantum authentication in TLS 1.3: A performance study. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[25] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-09, Internet Engineering Task Force, September 2023. Work in Progress.

[26] Tao Wang. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 152–167, 2020.

[27] Mozilla Wiki. Intermediate CA Preloading. https://wiki.mozilla.org/Security/CryptoEngineering/Intermediate_Preloading.