On Certificate Transparency Verification and Unlinkability of Websites Visited by Tor Users

Rasmus Dahlberg

Faculty of Health, Science and Technology

Computer Science

DOCTORAL THESIS | Karlstad University Studies | 2023:15

On Certificate Transparency Verification and Unlinkability of Websites Visited by Tor Users

Rasmus Dahlberg

DOCTORAL THESIS | Karlstad University Studies | 2023:15

On Certificate Transparency Verification and Unlinkability of Websites Visited by Tor Users

Rasmus Dahlberg

DOCTORAL THESIS

Karlstad University Studies | 2023:15

urn:nbn:se:kau:diva-94343

ISSN 1403-8099

ISBN 978-91-7867-372-8 (print)

ISBN 978-91-7867-373-5 (pdf)

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2023

WWW.KAU.SE

On Certificate Transparency Verification and Unlinkability of Websites Visited by Tor Users

Rasmus Dahlberg
Department of Mathematics and Computer Science

Abstract

Certificate Transparency is an ecosystem of logs, monitors, and auditors that hold certificate authorities accountable while issuing certificates. We show how the amount of trust that TLS clients and domain owners need to place in Certificate Transparency can be reduced, both in the context of existing gradual deployments and the largely unexplored area of Tor. Our contributions include improved third-party monitoring, a gossip protocol plugging into Certificate Transparency over DNS, an incrementally deployable gossip-audit model tailored for Tor Browser, and using certificates with onion addresses. The methods used range from proof sketches to Internet measurements and prototype evaluations. An essential part of our evaluation in Tor is to assess how the protocols used during website visits—such as requesting an inclusion proof from a Certificate Transparency log—affect unlinkability between senders and receivers. We find that most false positives in website fingerprinting attacks can be eliminated for all but the most frequently visited sites. This is because the destination anonymity set can be reduced due to how Internet protocols work: communication is observable and often involves third-party interactions. Some of the used protocols can further be subject to side-channel analysis. For example, we show that remote (timeless) timing attacks against Tor's DNS cache reliably reveal the timing of past exit traffic. The severity and practicality of our extension to website fingerprinting pose threats to the anonymity provided by Tor. We conclude that access to a so-called website oracle should be an assumed attacker capability when evaluating website fingerprinting defenses.

Keywords: Auditing, Certificate Transparency, DNS, Gossip, Side-Channels, Timing Attacks, Tor, Tor Browser, Website Fingerprinting, Website Oracles

Kring verifiering av Certificate Transparency samt länkbarhet av Tor-användare och besökta webbplatser

Rasmus Dahlberg
Institutionen för matematik och datavetenskap

Sammanfattning

Projektet *Certificate Transparency* är ett ekosystem av loggar, övervakare och granskare som håller certifikatutfärdare till svars för utfärdade webbcertifikat. Vi visar hur säkerheten kan höjas i ekosystemet för både domäninnehavare och TLS-klienter i nuvarande system samt som del av anonymitetsnätverket Tor. Bland våra större bidrag är förbättrad övervakning av loggarna, ett skvallerprotokoll som integrerats med DNS, ett skvaller- och granskningsprotokoll som utformats specifikt för Tors webbläsare och ett förslag på hur domännamn med adresser i Tor kan bli mer tillgängliga. De metoder som använts varierar från säkerhetsbevis till internetmätningar och utvärderingar av forskningsprototyper. En viktig del av vår utvärdering i Tor är att avgöra hur protokoll som används av webbläsare påverkar möjligheten att koppla ihop användare med besökta webbplatser. Detta inkluderar existerande protokoll samt nya tillägg för att verifiera om webbplatsers certifikat är transparensloggade. Våra resultat visar att i många fall kan falska positiva utslag filtreras bort vid mönsterigenkänning av Tor-användares krypterade trafik (eng: *website fingerprinting*). Orsaken är att besök till de flesta webbplatser kan uteslutas till följd av hur internetprotokoll fungerar: kommunikation är observerbar och involverar ofta interaktioner med tredjeparter. Vissa protokoll har dessutom sidokanaler som kan analyseras. Vi visar exempelvis att Tors DNS-cache kan undersökas med olika varianter av tidtagningsattacker. Dessa attacker är enkla att utföra över internet och avslöjar vilka domännamn som slagits upp vid angivna tidpunkter. De förbättrade mönsterigenkänningsattackerna mot webbplatser är realistiska och hotar därför Tors anonymitet. Vår slutsats är att framtida försvar bör utvärderas utifrån att angripare har tillgång till ett så kallat webbplatsorakel.

Nyckelord: Granskning, Certificate Transparency, DNS, Skvaller, Sidokanaler, Tidtagningsattacker, Tor, Tors webbläsare, Mönsterigenkänning,Webbplatsorakel

Acknowledgements

I am fortunate to be surrounded by people that helped me reach this milestone in one piece. First, I want to acknowledge that my brother, Victor, provided me with unlimited tutoring before my PhD studies. I would not be on my current path without him. Second, Sarah and Andreas, thank you for challenging and supporting me. You have been a determinantal part of my personal growth and well-being. Finally, Mom, thank you for being around whenever I need it.

Several doors were opened for me at Karlstad University. Stefan Alfredsson and Stefan Lindskog kick-started my undergraduate studies with programming feedback, increased-pace study plans, and the opportunity to be involved in the department. Tobias Pulls introduced me to computer security research. We have worked closely ever since, and *I strongly recommend him as an advisor and collaborator*. Simone Fischer-Hübner, Stefan Lindskog, Leonardo Martucci, and Tobias Pulls all provided sincere advice during my PhD studies. Many other individuals selflessly helped me forward. Some of them include Ala Sarah Alaqra, Linus Nordberg, Jenni Reuben, Fredrik Strömberg, and Paul Syverson.

I am further grateful for my collaborators: Matthew Finkel, Toke Høiland-Jørgensen, Andreas Kassler, Linus Nordberg, Tobias Pulls, Tom Ritter, Paul Syverson, and Jonathan Vestin. They are all driven individuals that I learned a lot from. I also appreciate the generous funding received from the Swedish Knowledge Foundation and the Swedish Foundation for Strategic Research.

Karlstad University , May 2, 2023 Rasmus Dahlberg

List of Acronyms

The introductory summary refrains from using acronyms to make it easily digested. However, a few acronyms are used without definition because their full versions are likely less familiar to most readers. These acronyms are:

DNS Domain Name System

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure

IP Internet Protocol

P4 Programming Protocol-independent Packet Processors

RFC Request For Comments

RIPE Réseaux IP Européens

RSA Rivest Shamir Adleman

TCP Transmission Control Protocol

TLS Transport Layer Security

XDP eXpress Data Path

List of Appended Papers

I. **Rasmus Dahlberg** and Tobias Pulls. Verifiable Light-Weight Monitoring for Certificate Transparency Logs. NordSec (2018).

II. **Rasmus Dahlberg**, Tobias Pulls, Jonathan Vestin, Toke Høiland-Jørgensen, and Andreas Kassler. Aggregation-Based Certificate Transparency Gossip. SECURWARE (2019).

III. **Rasmus Dahlberg**, Tobias Pulls, Tom Ritter, and Paul Syverson. Privacy-Preserving & Incrementally-Deployable Support for Certificate Transparency in Tor. PETS (2021).

IV. **Rasmus Dahlberg**, Paul Syverson, Linus Nordberg, and Matthew Finkel. Sauteed Onions: Transparent Associations from Domain Names to Onion Addresses. WPES (2022).

V. Tobias Pulls and **Rasmus Dahlberg**. Website Fingerprinting with Website Oracles. PETS (2020).

VI. **Rasmus Dahlberg** and Tobias Pulls. Timeless Timing Attacks and Preload Defenses in Tor's DNS Cache. USENIX Security (2023).

Comments on my Participation

Paper I I had the initial idea and conducted most of the work myself. Tobias mainly contributed with discussions that lead to the final design.

Paper II Andreas and Tobias had the initial idea of exploring the intersection between Certificate Transparency and programmable packet processors. I did most of the design and writing with feedback from Tobias, our RIPE Atlas measurements, and our performance benchmarks with Jonathan and Toke.

Paper III I had the initial idea and was the main driver to move the work forward, first in discussion with Tobias and then together with Tom and Paul.

Paper IV Paul, Linus, and I had the initial idea of exploring how onion addresses fit into Certificate Transparency. Paul and I did most of the writing. I implemented our monitor, Linus our search engine, Matt our web extension.

Paper V Tobias is the main author and conducted most of the work. I mainly contributed by coining the name *website oracle*, evaluating sources of real-world website oracles, and performing our non-simulated network experiments.

Paper VI Tobias and I collaborated closely from start to finish with the following exceptions. I did most implementation work. Volunteers from DFRI—a Swedish non-profit and non-partisan organization that promotes digital rights—operated our exit relays. Tobias did most DNS cache data analysis. Tobias also had the initial idea, which was refined with feedback from Roger Dingledine.

List of Other Contributions

Throughout and before my PhD studies, I also contributed to the following:

- The Sigsum Project. https://www.sigsum.org/ (2020-present).

  Sigsum is a free and open-source software project that brings transparency to signed checksums. The design is simple and uses a proactive gossip-audit model. I founded Sigsum with Linus Nordberg and Fredrik Strömberg while working at Mullvad VPN as a side-line occupation.

- Tobias Pulls and **Rasmus Dahlberg**. Steady: A Simple End-to-End Secure Logging System. NordSec (2018).

  Tobias was the main driver. I mostly contributed with design discussions and a security formalization.

- **Rasmus Dahlberg**, Tobias Pulls, and Roel Peeters. Efficient Sparse Merkle Trees: Caching Strategies and Secure (Non-)Membership Proofs. NordSec (2016).

  This work started with my Bachelor's thesis. I did most writing with input from Tobias and Roel. Tobias did our benchmarking experiments.

- **Rasmus Dahlberg** and Tobias Pulls. Standardized Syslog Processing: Revisiting Secure Reliable Data Transfer and Message Compression. Technical Report, Karlstad University (2016).

  I was the main driver. Tobias contributed with continuous feedback.

Contents

Introductory Summary

1 Introduction

The security posture of the Internet increased significantly throughout the last decade. For example, the cleaned-up and formally verified TLS 1.3 protocol that underpins HTTPS has been rolled-out gradually [44], the certificates that specify which public keys to use when bootstrapping a secure connection can be obtained for free and automatically [1], and web browsers have shifted from positive to negative security indicators in favor of security-by-default [109]. The use of end-to-end encryption has further become the norm with services such as DNS-over-HTTPS [43], virtual private networks [27], Tor [24], and secure messaging [101] gaining traction. In other words, the era of attackers that can passively snoop and actively tamper with unencrypted network traffic is over.

What will remain the same is the incentive for attackers to snoop and tamper with network traffic. Therefore, the focus is (and will likely continue to be) on circumventing protocols that add security and privacy as they are deployed in the real world. For example, there is a long history of certificate mis-issuance that allows attackers to impersonate websites and thus insert themselves as machines-in-the-middle ("MitM") without actually breaking TLS [17, 96]. Or, in the case of encrypted channels that are hard to intercept, instead analyzing traffic patterns to infer user activity like which website is being visited [14, 40, 41, 58, 75, 102]. The bad news is that attackers only need to find one vulnerability in a deployed protocol or its software. Sometimes, such vulnerabilities can be purchased by zero-day brokers like Zerodium [114].

To address an attack vector, it is common to add countermeasures that frustrate attackers and/or increase the risk involved while trying to exploit a system. A good example is how the certificate authority ecosystem evolved. For background, certificate authorities are trusted parties that validate domain names before issuing certificates that list their public keys. Web browsers are shipped with hundreds of trusted certificate authorities, which means that the resulting TLS connections cannot be more secure than the difficulty of hijacking the weakest-link certificate authority [17]. A proposal eventually deployed to mitigate this issue is Certificate Transparency: an ecosystem of public append-only logs that publishes all certificates so that any mis-issuance can be *detected* by monitors [54, 55]. These logs have a cryptographic foundation that holds them and the issuing certificate authorities *accountable*, at least in theory. In practice, the logs are essentially trusted parties that must act honestly due to how web browsers shape their policies to respect user privacy [3, 33, 64, 97].

The first objective of this thesis is to better understand the current limits of Certificate Transparency by proposing and evaluating improvements which *reduce* the amount of trust that needs to be placed in third-party monitors and logs. We make a dent in the problem of Certificate Transparency verification both generally and concretely in the context of Tor Browser, which unlike Google Chrome and Apple's Safari does not support Certificate Transparency yet. For context, Tor Browser is a fork of Mozilla's Firefox that (among other things) routes user traffic through the low-latency anonymity network Tor [24, 77]. As part of our pursuit to improve the status quo for Certificate

Transparency verification in Tor Browser, the second objective of this thesis is to evaluate how the protocols used during website visits affect unlinkability between senders (web browsers) and receivers (websites). Our evaluation applies to our addition of Certificate Transparency and other protocols already in use, e.g., DNS, real-time bidding [110], and certificate revocation checking [85].

The remainder of the introductory summary is structured as follows. Section 2 introduces background that will help the reader understand the context and preliminaries of the appended papers. Section 3 defines our research questions and overall objective. Section 4 provides an overview of our research methods. Section 5 describes our contributions succinctly. Section 6 summarizes the appended papers that are published in NordSec (Paper I), SECURWARE (Paper II), PETS (Paper III and V), WPES (Paper IV), and USENIX Security (Paper VI). Section 7 positions our contributions with regard to related work. Section 8 concludes and briefly discusses future work.

2 Background

This section introduces background on Certificate Transparency and Tor.

2.1 Certificate Transparency

The web's public-key infrastructure depends on certificate authorities to issue certificates that map domain names to public keys. For example, the certificate of `www.example.com` is issued by DigiCert and lists a 2048-bit RSA key [87]. The fact that DigiCert signed this certificate means that they claim to have verified that the requesting party is really `www.example.com`, typically by first ensuring that a specified DNS record can be uploaded on request [11]. If all certificate authorities performed these checks correctly and the checks themselves were fool-proof, a user's browser could be sure that any certificate signed by a certificate authority would list a verified public key that can be used for authentication when connecting to a website via TLS. Unfortunately, there are hundreds of trusted certificate authorities and a long history of issues surrounding their operations in practice [8, 17, 96]. One of the most famous incidents took place in 2011: an attacker managed to mis-issue certificates from DigiNotar to intercept traffic towards Google and others in Iran [45]. The astonishing part is that this incident was first detected *seven weeks later*.

Certificate Transparency aims to facilitate *detection* of issued certificates, thus holding certificate authorities *accountable* for any certificates that they mis-issue [54, 55]. The basic idea is shown in Figure 1. In addition to regular validation rules, browsers ensure certificates are included in a public append-only Certificate Transparency *log*. This allows anyone to get a concise view of all certificates that users may encounter, including domain owners like Google who can then see for themselves whether any of the published certificates are mis-issued. The parties inspecting the logs are called *monitors*. Some monitors mirror all log entries [86], while others discard most of them in pursuit of

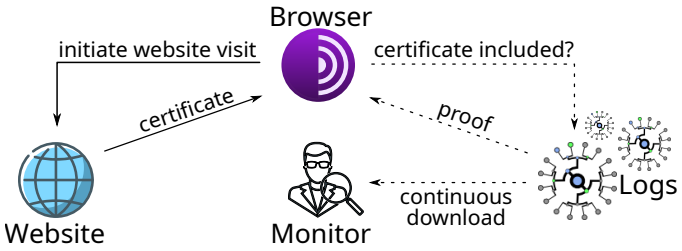

Figure 1: The idea of Certificate Transparency. Certificates encountered by users must be included in a public log so that monitors can detect mis-issuance.

finding matches for pre-defined criteria like `*.example.com` [95]. Another option is subscribing to certificate notifications from a trusted third-party [34].

What makes Certificate Transparency a significant improvement compared to the certificate authority ecosystem is that the logs stand on a cryptographic foundation that can be verified. A log can be viewed as an append-only tamper-evident list of certificates. It is efficient[1] to prove cryptographically that a certificate is in the list, and that a current version of the list is append-only with regard to a previous version (i.e., no tampering or reordering).[2] These properties follow from using a Merkle tree structure that supports *inclusion* and *consistency* proofs [19, 28, 55, 67]. The reader only needs to know that these proofs are used to reconstruct a log's Merkle tree head, often referred to as a *root hash*. It is a cryptographic hash identifying a list of certificates uniquely in a tree data structure. The logs sign root hashes with the number of entries and a timestamp to form *signed tree heads*. So, if an inconsistency is discovered, it cannot be denied. Log operators are therefore held accountable for maintaining the append-only property. A party that verifies the efficient transparency log proofs without downloading all the logs is called an *auditor*.

A log that signs two inconsistent tree heads is said to perform a *split-view*. To ensure that everyone observes the same append-only logs, all participants of the Certificate Transparency ecosystem must engage in a *gossip protocol* [16, 72]. In other words, just because Alice observes an append-only log, it is not necessarily the *same append-only log* that Bob observes. Therefore, Alice and Bob must exchange signed tree heads and verify consistency to assert that the log operators play by the rules and only append certificates. Without a secure gossip protocol, log operators would have to be trusted blindly (much like certificate authorities before Certificate Transparency). RFC 6962 defers the specification of gossip [55], with little or no meaningful gossip deployed yet.

Rolling out Certificate Transparency without breakage on the web is a challenge [98]. Certificates must be logged, associated proofs delivered to end-user software, and more. One solution RFC 6962 ultimately put forth was the introduction of *signed certificate timestamps*. A signed certificate timestamp is a log's *promise* that a certificate will be appended to the log within a *maximum*

[1] Efficient refers to space-time complexity $O(\log(n))$, where n is the number of log entries.
[2] Interested readers can refer to our Merkle tree and proof technique introduction online [21].

merge delay (typically 24 hours). Verifying if a log holds its promise is usually called *auditing*. Certificate authorities can obtain signed certificate timestamps and embed them in their final certificates by logging a *pre-certificate*. As such, there is no added latency from building the underlying Merkle tree and no need for server software to be updated (as the final certificate contains the information needed). The current policy for Google Chrome and Apple's Safari is to reject certificates with fewer than two signed certificate timestamps [3, 33]. How to request an inclusion proof for a promise without leaking the user's browsing history to the log is an open problem [64]. In other words, asking for an inclusion proof trivially reveals the certificate of interest to the log.

Other than embedding signed certificate timestamps in certificates, they can be delivered dynamically to end-users in TLS extensions and stapled certificate status responses. For example, Cloudflare uses the TLS extension delivery method to recover from log incidents without their customers needing to acquire new certificates [100]. Several log incidents have already happened in the past, ranging from split-views [6, 92, 93] to broken promises of timely logging [29, 37, 5, 91] and potential key compromise [84]. These are all good *scares* motivating continued completion of Certificate Transparency in practice.

In summary, the status quo is for web browsers to require at least two signed certificate timestamps before accepting a certificate as valid. Merkle tree proofs are not verified. Gossip is not deployed. The lack of a reliable *gossip-audit model* means that the logs are largely trusted parties.[3] We defer discussion of related work in the area of gossip-audit models until Section 7.

2.2 Tor

The Tor Project is a 501(c)(3) US nonprofit that advances human rights and defends privacy online through free software and open networks [79]. Some of the maintained and developed components include Tor Browser and Tor's relay software. Thousands of volunteers operate relays as part of the Tor network, which routes the traffic of millions of daily users with low latency [61]. This frustrates attackers like Internet service providers that may try linking *who is communicating with whom* from their local (non-global) vantage points [24].

Usage of Tor involves tunneling the TCP traffic of different destinations (such as all flows associated with a website visit to `example.com`) in fixed-size *cells* on independent *circuits*. A circuit is built through a guard, a middle, and an exit relay. At each hop of the circuit, one layer of symmetric encryption is peeled off. The used keys are ephemeral and discarded together with all other circuit state after at most 10 minutes (the maximum circuit lifetime). This setup allows guard relays to observe users' IP addresses but none of the destination traffic. In contrast, exit relays can observe destination traffic but no user IP addresses. The relays used in a circuit are determined by Tor's end-user software. Such path selection is randomized and bandwidth-weighted but starts

[3] Historical remark: the lack of verification led Google to require that all certificates be disclosed in at least one of their logs to validate [97]. The so-called *one-Google log requirement* was recently replaced. Google Chrome instead interacts with Google's trusted auditor. See Section 7.

with a largely static guard set to protect users from *eventually* entering the network from a relay an attacker volunteered to run.

Tor's *consensus* lists the relays that make up the network. As the name suggests, it is a document agreed upon by a majority of trusted *directory authorities*. Five votes are currently needed to reach a consensus. Examples of information added to the Tor consensus include tunable network parameters and uploaded relay descriptors with relevant metadata, e.g., public key, available bandwidth, and exit policy. Each relay in the consensus is also assigned different flags based on their configuration and observed performance, e.g., `Guard`, `MiddleOnly`, `Fast`, `Stable`, and `HSDir`. The latter means that the relay is a *hidden service directory*, which refers to being part of a distributed hash table that helps users lookup *onion service introduction points*.

An onion service is a self-authenticated server identified by its public key. Onion services are only reachable through the Tor network. Users that are aware of a server's *onion address* can consult the distributed hash table to find its introduction points. To establish a connection, a user builds a circuit to a *rendezvous point*. A request is then sent to one of the current introduction points, which informs the onion service that it may build its own circuit to meet the user at their rendezvous point. In total, six relays are traversed while interacting with an onion service. This setup allows not only the sender but also the receiver to be anonymous. The receiver also benefits from a large degree of censorship resistance as the server location may be hidden. The main drawback of onion services is that their non-mnemonic names are hard to discover and remember. Some sites try to overcome this by setting their onion addresses in *onion location* HTTP headers or HTML attributes [80].

Many users use Tor Browser to connect to the Tor network. In addition to routing traffic as described above, Tor Browser ships with privacy-preserving features like first-party isolation to not share any state across different origins, settings that frustrate browser fingerprinting, and *disk-avoidance* to not store browsing-related history as well as other identifying information to disk [77]. Tor Browser is a fork of Mozilla's Firefox. Unfortunately, neither Firefox nor Tor Browser supports any form of Certificate Transparency. Conducting undetected machine-in-the-middle attacks against Tor users is thus relatively straightforward: compromise or coerce the weakest-link certificate authority, then volunteer to operate an exit relay and intercept network traffic. Such interception has previously been found with self-signed certificates [113].

While global attackers are not within Tor's threat model, it is in scope to guard against various local attacks [24]. For example, the intended attacker may passively observe a small fraction of the network and actively inject their own packets. Figure 2 shows the typical attacker setting of *website fingerprinting*, where the attacker observes a user's entry traffic with the goal of inferring which website was visited solely based on analyzing encrypted traffic [14, 40, 41, 58, 75, 102]. Website fingerprinting attacks are evaluated in the *open-world* or *closed-world* settings. In the closed-world setting, the attacker monitors (not to be confused with Certificate Transparency monitoring) a fixed list of websites. A user visits one of the monitored sites, and the attacker needs to

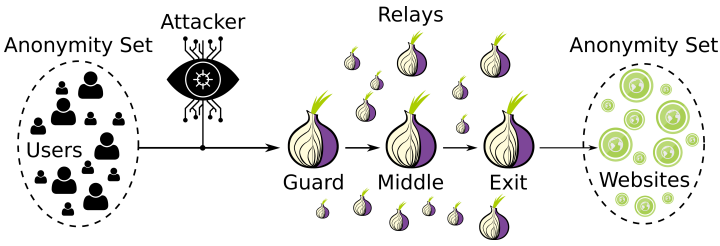

Figure 2: The setting of a website fingerprinting attack. A local passive attacker analyzes a user's encrypted network traffic as it enters the network. The goal is to infer which website is visited. (Figure reprinted from Paper V.)

determine which one. The open-world setting is the same as the closed-world setting, except that the user may also visit unmonitored sites. The practicality of website fingerprinting attacks is up for debate, e.g., ranging from challenges handling false positives to machine-learning dataset drift [15, 47, 76, 111].

In summary, Tor is a low-latency anonymity network often accessed with Tor Browser. Among the threats that Tor aims to protect against are local attackers that see traffic as it enters or leaves the network (but not both at the same time all the time). A website fingerprinting attack is an example of a passive attack that operates on entry traffic. A machine-in-the-middle attack is an example of an active attack that typically operates on exit traffic. Discussion of related work in the area of website fingerprinting is deferred until Section 7.

3 Research Questions

The overall research objective spans two different areas: transparency logs and low-latency anonymity networks. We aim to reduce trust assumptions in transparency log solutions and to apply such solutions in anonymous settings for improved security and privacy. We defined the following research questions to make this concrete in Certificate Transparency and Tor, the two ecosystems with the most history and dominant positions in their respective areas.

1. *Can trust requirements in Certificate Transparency be reduced in practice?*

   Transparency logs have a cryptographic foundation that supports efficient verification of inclusion and consistency proofs. Such proofs are useful to reduce the amount of trust that is placed in the logs. The roll-out of Certificate Transparency has yet to start using these proofs, and to employ a gossip protocol that ensures the same append-only logs are observed. Part of the challenge relates to privacy concerns as parties interact with each other, as well as deploying gradually without breakage.

   We seek practical solutions that reduce the trust requirements currently placed in the logs and third-party monitors while preserving user privacy.

2. *How can authentication of websites be improved in the context of Tor?*

   Tor Browser has yet to support Certificate Transparency to facilitate detection of hijacked websites. This includes HTTPS sites but also onion services that may be easier to discover reliably with more transparency.

   We seek incremental uses of Certificate Transparency in Tor that preserve user privacy while engaging in new verification protocols to reduce trust.

3. *How do the protocols used during website visits affect unlinkability between Tor users and their destination websites?*

   Several third-parties become aware of a user's browsing activities while a website is visited. For example, DNS resolvers and certificate status responders may be consulted for domain name resolution and verification of if a certificate has been revoked. Fetching an inclusion proof from a Certificate Transparency log would reveal the same type of information.

   We seek to explore how unlinkability between Tor users and their exit destinations is affected by the multitude of protocols used during website visits. The considered setting is the same as in website fingerprinting, except that the attacker may take additional passive and active measures. For example, the attacker may volunteer to run a Certificate Transparency log (passive) or inject carefully-crafted packets into Tor (active).

4 Research Methods

We tackle privacy and security problems in the field of computer science [23, 26]. Our work is applied, following the scientific method for security and experimental networking research. *Exactly* what it means to use the scientific method in these areas is up for debate [7, 39]. However, at a glance, it is about forming precise and consistent theories with falsifiable predictions *as in other sciences* except that the objects of study are *information systems in the real world*.

A prerequisite to formulating precise, consistent, and falsifiable theories is that there are few implicit assumptions. Therefore, scientific security research should be accompanied by definitions of security goals and attacker capabilities: what does it mean that the system is secure, and what is the attacker (not) allowed to do while attacking it [51]? Being explicit about the overall *setting* and *threat model* is prevalent in formal security work like cryptography, where an abstract (mathematical) model is used to show that security can be *proved* by reducing to a computationally hard problem (like integer factorization) or a property of some primitive (like the collision resistance of a hash function) [50]. It is nevertheless just as crucial in less formal work that deals with security of systems in the real (natural) world—the exclusive setting of the scientific method—which usually lends itself towards break-and-fix cycles in light of new observations. Where to draw the line between *security work* and *security research* is not trivial. However, a few common *failures* of past "security research" include not bringing observations in contact with theory, not making claims and assumptions explicit, or simply relying on unfalsifiable claims [39].

While deductive approaches (like formal reduction proofs) are instrumental in managing complexity and gaining confidence in different models, more than these approaches are required as a model's *instantiation* must also be secure [51]. It is common to complement abstract modeling with *real-world measurements* as well as *systems prototyping and evaluations* [7]. Real-world measurements measure properties of deployed systems like the Internet, the web, and the Tor network. For example, a hypothesis in a real-world measurement could be that (non-)Tor users browse according to the same website popularity distribution. Sometimes these measurements involve the use of research prototypes, or the research prototypes themselves become the objects of study to investigate properties of selected system parts (say, whether a packet processor with new features is indistinguishable from some baseline as active network attackers adaptively inject packets of their choosing). If it is infeasible, expensive, or unsafe (see below) to study a real-world system, a simulation may be studied instead. The downside of simulation is that the model used may not be a good approximation of the natural world, similar to formal cryptographic modeling.

The appended papers use all of the above approaches to make claims about security, privacy, and performance in different systems, sometimes with regard to an abstract model that can be used as a foundation in the natural world to manage complexity. Paper I contains a reduction proof sketch to show reliance on standard cryptographic assumptions. Paper V extends past simulation setups to show the impact of an added attacker capability. Meanwhile, Paper III models part of the Tor network with mathematical formulas to estimate performance overhead. All but Paper IV contain real-world measurements relating to Internet infrastructure, websites, certificates, Tor, or practical deployability of our proposals. All but Paper III contain research prototypes with associated evaluations, e.g., performance profiling, as well as corroborating or refuting our security definitions in experimental settings. All papers include discussions of security and privacy properties as well as their limitations and strengths in the chosen settings (where assumptions are explicit and threat models motivated).

Throughout our experiments, we strived to follow best practices like documenting the used setups, making datasets and associated tooling available, reducing potential biases by performing repeated measurements from multiple different vantage points, and discussing potential biases (or lack thereof) [7]. We also interacted with Tor's research safety board [81] to discuss the ethics and safety of our measurements in Paper VI, and refrained from measuring real (i.e., non-synthetic) usage of Tor whenever possible (Papers III and V). Finally, the uncovered bugs and vulnerabilities in Papers V–VI were responsibly disclosed to the Tor project. This included suggestions on how to move forward.

5 Contributions

The main contributions of this thesis are listed below. An overview of how they relate to our research questions and appended papers is shown in Figure 3.

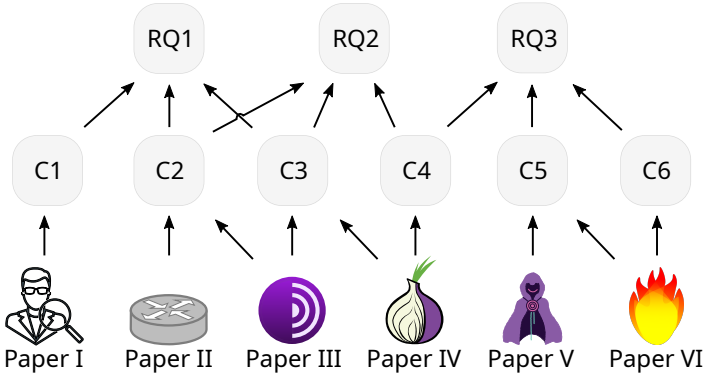

Figure 3: Overview of appended papers, contributions, and research questions.

1. *Reduced trust in third-party monitoring with a signed tree head extension that shifts trust from non-cryptographic certificate notifications to a log's gossip-audit model (or if such a model does not exist yet, the logs themselves).*

   Paper I applies existing cryptographic techniques for constructing static and lexicographically ordered Merkle trees so that certificates can be wild-card filtered on subject alternative names with (non-)membership proofs. This building block is evaluated in the context of Certificate Transparency, including a security sketch and performance benchmarks.

2. *Increased probability of split-view detection by proposing gossip protocols that disseminate signed tree heads without bidirectional communication.*

   Paper II explores aggregation of signed tree heads at line speed in programmable packet processors, facilitating consistency proof verification on the level of an entire autonomous system. Such verification can be indistinguishable from an autonomous system without any split-view detection to achieve herd immunity, i.e., protection without aggregation. Aggregation at 32 autonomous systems can protect 30-50% of the IPv4 space. Paper III explores signed tree heads in Tor's consensus. To reliably perform an undetected split-view against log clients that have Tor in their trust root, a log must collude with a majority of directory authorities.

3. *Improved detectability of website hijacks targeting Tor Browser by proposing privacy-preserving and gradual roll-outs of Certificate Transparency in Tor.*

   Paper III explores adoption of Certificate Transparency in Tor Browser with signed certificate timestamps as a starting point, then leveraging the decentralized network of relays to cross-log certificates before ultimately verifying inclusion proofs against a single view in Tor's consensus. The design is probabilistically secure with tunable parameters that result in modest overheads. Paper IV shows that Certificate Transparency logging of domain names with associated onion addresses helps provide forward censorship-resistance and detection of unwanted onion associations.

4. *An extension of the attacker model for website fingerprinting that provides attackers with the capability of querying a website oracle.*

   A website oracle reveals whether a monitored website was (not) visited by any network user during a specific time frame. Paper V defines and simulates website fingerprinting attacks with website oracles, showing that most false positives can be eliminated for all but the most frequently visited websites. A dozen sources of real-world website oracles follow from the protocols used during website visits. We enumerate and classify those sources based on ease of accessibility, reliability, and coverage. The overall analysis includes several Internet measurements.

5. *Remotely-exploitable probing-attacks on Tor's DNS cache that instantiate a real-world website oracle without any special attacker capabilities or reach.*

   Paper V shows that timing differences in end-to-end response times can be measured to determine whether a domain name is (not) cached by a Tor relay. An estimated true positive rate of 17.3% can be achieved while trying to minimize false positives. Paper VI improves the attack by exploiting timeless timing differences that depend on concurrent processing. The improved attack has no false positives or false negatives. Our proposed bug fixes and mitigations have been merged in Tor.

6. *A complete redesign of Tor's DNS cache that defends against all (timeless) timing attacks while retaining or improving performance compared to today.*

   Paper VI suggests that Tor's DNS cache should only share the same preloaded domain names across different circuits to remove the remotely-probable state that reveals information about past exit traffic. A network measurement with real-world Tor relays shows which popularity lists are good approximations of Tor usage and, thus, appropriate to preload. Cache-hit ratios can be retained or improved compared to today's Tor.

6 Summary of Appended Papers

The appended papers and their contexts are summarized below. Notably, all papers are in publication-date order except that Paper V predates Papers III–IV.

Paper I – Verifiable Light-Weight Monitoring for Certificate Transparency Logs

An often overlooked part of Certificate Transparency is that domain owners are expected to inspect the logs for mis-issued certificates continuously. The cost and required expertise to do so have led to the emergence of third-party monitoring services that notify domain owners of newly issued certificates that they subscribe to. For example, one may subscribe to email notifications whenever a certificate is issued for `*.example.com`. One downside of such third-party monitoring is that these notification services become trusted parties

with little or no accountability with regard to omitted certificate notifications. We show how to add this accountability and tie it to the gossip-audit model employed by the Certificate Transparency ecosystem by proposing verifiable light-weight monitoring. The idea is for logs to batch appended certificates into an additional data structure that supports *wild-card (non-)membership proofs*. As a result, third-party monitors can prove cryptographically that they did not omit any certificate notifications selectively. Our experimental performance evaluation shows that overhead can be tuned to be small for all involved parts.

Paper II – Aggregation-Based Certificate Transparency Gossip

Another often overlooked part of Certificate Transparency is that monitors and end-users who browse websites must observe the same append-only logs. For example, if the same append-only logs are not observed, an end-user may connect to a website that serves a mis-issued certificate that no monitor will discover. This would largely defeat the purpose of public logging, which is why RFC 6962 specifies that multiple gossip protocols should be defined separately in the future. We define one such protocol that plugs into the (at the time current) idea of having end-users interact with the logs through DNS. Our work is exploratory, using recent advancements in programmable packet processors that allow turning routers, switches, and network interface cards into *aggregators* of tree heads that the logs signed and transmitted in plaintext via DNS. The aggregated tree heads are then used as a reference while challenging the logs to prove consistency, thus protecting entire vantage points from undetected split views. A different network path (like Tor) can be used to break out of a local vantage point to increase the likelihood of global consistency. If the security definition for *aggregation indistinguishability* is satisfied, vantage points without an aggregator may also receive protection due to herd immunity. Our P4 and XDP prototypes satisfy the notion of aggregation indistinguishability at line-rate with regard to throughput. Prevalent vantage points to roll out aggregation-based gossip include autonomous systems and Internet exchange points that route the traffic of many users. Our RIPE Atlas measurements show that 32 autonomous systems could protect 30-50% of the IPv4 space from undetected split views. End-users merely need to use plaintext DNS for opt-in.

Paper III – Privacy-Preserving & Incrementally-Deployable Support for Certificate Transparency in Tor

One deployment challenge of Certificate Transparency is to ensure that monitors and end-users are engaged in gossip-audit protocols. This is particularly difficult for end-users because such engagement can harm privacy. For example, verifying that a certificate is included by fetching an inclusion proof from a log reveals which website was visited. We propose a gradual roll-out of Certificate Transparency in Tor Browser that preserves privacy *due to* and *how we use* the anonymity network Tor. The complete design holds log operators accountable for certificates they promise to append by having Tor relays fetch inclusion

proofs against the same view agreed upon by directory authorities in Tor's consensus. Found issues (if any) are reported to trusted auditors. The incremental design side-steps much of the practical deployment effort by replacing the audit-report pattern with cross-logging of certificates in independent logs, thus assuming that at least one log is honest as opposed to no log in the complete design. All Tor Browser needs to do is verify log signatures and then submit the encountered certificates to randomly selected Tor relays. Such submissions are probabilistic to balance performance against the risk of eventual detection of log misbehavior. Processing of the submitted certificates is also randomized to reduce leakage of real-time browsing patterns, something Tor Browser cannot do on its own due to criteria like disk avoidance and the threat model for wanting Certificate Transparency in the first place. We provide a security sketch and estimate performance overhead based on Internet measurements.

Paper IV – Sauteed Onions: Transparent Associations from Domain Names to Onion Addresses

Many prominent websites are also hosted as Tor onion services. Onion services are identified by their public keys and subject to onion routing, thus offering self-authenticated connections and censorship resistance. However, the non-mnemonic names are a limitation due to being hard to discover and remember. We explore how certificates with onion addresses may improve the status quo by proposing sauteed onions, *transparent associations from domain names to onion addresses* with the help of Certificate Transparency logs. The idea is to extend a website's regular certificate with an associated onion address. This makes it possible to offer certificate-based onion location that is no less targeted than the HTTPS connection facilitating the discovery, as well as name-to-onion search engines that use the append-only logs for verifiable population of their databases. The achieved goals are consistency of available onion associations, improved third-party discovery of onion associations, and forward censorship-resistance. To be discovered, sites must opt-in by obtaining a sauteed onion certificate. Our prototypes for certificate-based onion location and third-party search engines use an existing backward-compatible format. We discuss this trade-off and note that a certificate extension may be used in the future.

Paper V – Website Fingerprinting with Website Oracles

One of the properties Tor aims to provide against local network attackers is unlinkability between end-users (sender anonymity set) and their destinations on the Internet (receiver anonymity set). A website fingerprinting attack aims to break anonymity in this model by inferring which website an identifiable end-user is visiting based only on the traffic entering the Tor network. We extend the attacker model for website fingerprinting attacks by introducing the notion of *website oracles*. A website oracle answers the following question: was website w visited during time frame t? In other words, the attacker can query the receiver anonymity set for websites that were (not) visited. Our simulations show that augmenting past website fingerprinting attacks to include website

oracles significantly reduces false positives for all but the most popular websites, e.g., to the order of 10^{-6} for classifications around Alexa top-10k and much less for the long tail of sites. Further, some earlier website fingerprinting defenses are largely ineffective in the (stronger) attacker model that includes website oracles. We discuss a dozen real-world website oracles ranging from centralized access logs to widely accessible real-time bidding platforms and DNS caches, arguing that they are inherent parts of the protocols used to perform website visits. Therefore, access to a website oracle should be an assumed attacker capability when evaluating which website fingerprinting defenses are effective.

Paper VI – Timeless Timing Attacks and Preload Defenses in Tor's DNS Cache

Tor relays cache resolved domains with constant time-to-live values not to reveal information about past exit traffic while boosting performance. We show that this caching strategy and its implementation in the live Tor network can be exploited by a *timeless timing attack* that leaks if a domain is (not) cached. Further, the time that a domain was inserted into the cache can be inferred by repeated probes. Our attack prototype's experimental evaluation in real conditions shows that there are neither false positives nor false negatives (10M repetitions). Thus, it is useful for instantiating a real-world website oracle without requiring any special attacker capabilities or reach (just a modest computer that can create a Tor circuit). One of our mitigations has been merged in Tor: probabilistic time-to-live values that make the time-of-insertion fuzzy. Long-term, Tor's DNS cache could be redesigned to *preload* the same domains at all exits. Such preloading would eliminate all (timeless) timing attacks in Tor's DNS cache because the same domains would always be (un)cached across different circuits. To retain performance within the same circuit, we propose that the preloaded domains should be complemented by a dynamic same-circuit cache that is not shared across circuits. Our four-month-long DNS cache measurement at two 100 Mbit/s exit relays informs on today's baseline performance. It is compared to a preloaded DNS cache based on different variations of three popularity lists: Alexa, Tranco, and Umbrella. A preloaded DNS cache can be as performant as today with similar resource usage or significantly improve cache-hit ratios by 2-3x. However, the increased cache-hit ratios have the cost of modest increases in memory and resolver load.

7 Related Work

This section positions the appended papers with regard to related work. For Certificate Transparency, this includes approaches towards signed certificate timestamp verification, gossip, and the problem of monitoring the logs. The related work with regard to Tor is focused on the practicality of website fingerprinting attacks and prior use of side-channels (such as timing attacks).

7.1 Certificate Transparency Verification

Approaches that fetch inclusion proofs have in common that they should preserve privacy by not revealing the link between users and visited websites. Eskandarian *et al.* mention that Tor could be used to overcome privacy concerns; however, it comes at the cost of added infrastructure requirements [31]. Lueks and Goldberg [59] and Kales *et al.* [49] suggest that logs could provide inclusion proofs using multi-server private information retrieval. This requires a non-collusion assumption while also adding significant overhead. Laurie suggests that users can fetch inclusion proofs via DNS as their resolvers already learned the destination sites [53]. While surveying signed certificate timestamp auditing, Meiklejohn *et al.* point out that Certificate Transparency over DNS may have privacy limitations [64]. For example, the time of domain lookups and inclusion proof queries are detached. Paper II uses Laurie's approach as a premise while proposing a gossip protocol. Paper III applies Certificate Transparency in a context where Tor is not additional infrastructure (Tor Browser).

Several proposals try to avoid inclusion proof fetching altogether. Dirksen *et al.* suggest that all logs could be involved in the issuance of a signed certificate timestamp [25]. This makes it harder to violate maximum merge delays without detection but involves relatively large changes to log deployments. Nordberg *et al.* suggest that signed certificate timestamps can be handed back to the origin web servers on subsequent revisits [72], which has the downside of assuming that machine-in-the-middle attacks eventually cease for detection. Nordberg *et al.* [72] as well as Chase and Meiklejohn [13] suggest that some clients/users could collaborate with a trusted auditor. Stark and Thompson describe how users can opt-in to use Google as a trusted auditor [99]. This approach was recently replaced by opt-out auditing that cross-checks a fraction of signed certificate timestamps with Google using k-anonymity [22]. Henzinger *et al.* show how such k-anonymity can be replaced with a single-server private information retrieval setup that approaches the performance of prior multi-server solutions [38]. None of the latter two proposals provide a solution for privately reporting that a log may have violated its maximum merge delay because the trusted auditor is assumed to know about all signed certificate timestamps. Eskandarian *et al.* show how to prove that a log omitted a certificate privately [31]. However, they use an invalid assumption about today's logs being in strict timestamp order [64]. Paper III suggests that Tor Browser could submit a fraction of signed certificate timestamps to randomly selected Tor relays. These relays perform further auditing on Tor Browser's behalf: much like a trusted auditor, except that no single entity is running it.

Merkle trees fix log content—not promises of logging. Therefore, inclusion proof fetching by users or their trusted parties must be accompanied by consistency verification and gossip to get a complete gossip-audit model [55]. Chuat *et al.* suggest that users and web servers can pool signed tree heads, gossiping about them as they interact [16]. Nordberg *et al.* similarly suggest that users can pollinate signed tree heads as they visit different web servers [72]. Hof and Carle suggest that signed tree heads could be cross-logged to make all logs intertwined [42]. Gunn *et al.* suggest multi-path fetching of signed tree

heads [36], which may make persistent split-views hard depending on the used multi-paths. Syta *et al.* suggest that independent witnesses could cosign the logs using threshold signatures [103]. Smaller-scale versions of witness cosigning received attention in industry [18, 65], and generally in other types of transparency logs as well [60]. Larger browser vendors could decide to push the same signed tree heads to their users, as proposed by Sleevi and Messeri [94]. Paper II uses the operators of network vantage points for aggregating and verifying signed tree heads to provide their users with gossip-as-a-service, however assuming plaintext DNS traffic and a sound signed tree head frequency as defined by Nordberg *et al.* [72]. We used the multi-path assumptions of Gunn *et al.* to break out of local vantage points. In contrast, Paper III ensures that the same logs are observed in the Tor network by incorporating signed tree heads into Tor's consensus (thus making directory authorities into witnesses).

Li *et al.* argue that it would be too costly for most domains to run a monitor [57].[4] Similar arguments have been raised before, and lead to alternative data structures that could make monitoring more efficient than today's overhead [30, 66, 104]. Paper I falls into this category, as the root of an additional static lexicographically-ordered Merkle tree is added to a log's signed tree heads to encode batches of included certificates. The downside is that a non-deployed signed tree head extension is assumed [56], as well as a tree head frequency similar to those described by Nordberg *et al.* [72] to get efficiency in practise.

Paper IV uses a Mozilla Firefox web extension to verify embedded signed certificate timestamps in Tor Browser. Such verification is similar to the gradual deployments of Certificate Transparency in other browsers [97, 98], and the starting point to improve upon in Papers II–III. Moreover, the use of Certificate Transparency to associate human-meaningful domain names with non-mnemonic onion addresses (as in Paper IV) is one of many proposals for alternative naming systems and onion search solutions [48, 69, 73, 80, 88, 108].

7.2 Website Fingerprinting and Side-Channels

Several researchers outline how past website fingerprinting attacks have been evaluated in unrealistic conditions [47, 76, 111]. This includes not accounting for the size of the open-world setting, failing to keep false positive rates low enough to be useful, assuming that homepages are browsed one at a time, how to avoid dataset drift, and training classifiers on synthetic network traces. While some of these challenges were addressed [15, 111], the question of how to deal with false positives remains open. Papers V–VI make a significant dent in this problem by providing evidence that the website fingerprinting attacker model could be made *stronger* to capture *realistic real-world capabilities* that eliminate most false positives around Alexa top-10k and the long tail of unpopular sites.

Others have evaluated traffic analysis attacks against Tor beyond the website fingerprinting setting. On one side of the spectrum are end-to-end correlation/confirmation attacks that typically consider a global passive attacker that observes all network traffic [46, 71, 74, 83]. Such strong attackers are not

[4] Whether the third-party monitors in this study misbehaved or not can be questioned [4].

within the scope of Tor [24]. On the other side of the spectrum are local attackers that see a small fraction of the network, typically in a position to observe a user's encrypted entry traffic (Figure 2). Many have studied those *weak attacks* in lab settings where, e.g., advances in deep learning improved the accuracy significantly [63, 82, 90]. Others have focused on improved attacks that are *active* in the Tor network from their own local vantage points [12, 68, 70], which is similar to the techniques in Papers V–VI. Greschbach *et al.* show that an attacker who gains access to (or traffic to [89]) commonly used DNS resolvers like Google's `8.8.8.8` get valuable information to improve both end-to-end correlation and website fingerprinting attacks [35]. Paper V generalizes the attacker capability they uncovered by allowing the attacker to query Tor's receiver anonymity set with a website oracle of time-frame t. It is further shown that it is possible to instantiate such an abstraction in the real-world while *staying within Tor's threat model*. In other words, the attacker is still local but may employ passive and active measures to narrow down the receiver anonymity set. Paper III proposes Certificate Transparency verification that gives log operators website oracle access. Tor's directory authorities tune t.

Website oracles exist because Tor is designed for anonymity—not unobservable communication [78]. The instantiation of a real-world website oracle is either a direct result of observing network flows from the protocols used during website visits, or due to state of these network flows being stored and inferable. Inferring secret system state is widely studied in applied cryptography and hardware architecture [2, 32, 52, 62, 105, 107], where the goal is usually to determine a key, decrypt a ciphertext, forge a message, or similar using side-channels. A side-channel can be local or remote and ranges from analysis of power consumption to cache states and timing differences. There is a long history of remote timing attacks that are practical [9, 10, 20, 112]. A recent improvement in this area that is relevant for Tor is timeless timing attacks, which exploit concurrency and message reordering to eliminate network jitter [106]. Paper V demonstrates a remote timing attack against Tor's DNS cache that achieves up to 17.3% true positive rates while minimizing false positives. Paper VI instead uses a remote timeless timing attack with no false positives, no false negatives, and a small time-frame t. This approaches an ideal website oracle without special attacker capabilities or reach into third-parties.

8 Conclusions and Future Work

Throughout the thesis, we contributed to the understanding of how trust requirements in Certificate Transparency can be reduced. Efficient and reliable monitoring of the logs is easily overlooked. If the broader ecosystem achieves monitoring through third-parties, they should be subject to the same scrutiny as logs. We proposed a solution that makes it hard for third-party monitors to provide subscribers with selective certificate notifications. We also proposed a gossip-audit model that plugs into interacting with the logs over DNS by having programmable packet processors verify that the same append-only logs are observed. Avoiding the addition of complicated verification logic into

end-user software is likely a long-term win because it reduces the number of moving parts. In other words, simple gossip-audit models will be much easier to deploy in the wide variety of end-user software that embeds TLS clients.

We also contributed to the understanding of how Certificate Transparency can be applied in the context of Tor Browser. Compared to a regular browser, this results in a different setting with its own challenges and opportunities. On the one hand, Tor Browser benefits from the ability to preserve privacy due to using the anonymity network Tor. On the other hand, data relating to website visits cannot be persisted to disk (such as signed certificate timestamps blocked by maximum merge delays). Our incrementally-deployable proposal keeps the logic in Tor Browser simple by offloading all Certificate Transparency verification to randomly selected Tor relays. The design is complete because mis-issued certificates can eventually reach a trusted auditor who acts on incidents. In addition to proposing Certificate Transparency in Tor Browser, we also explored how certificates with onion addresses may improve the association of domain names with onion addresses. Such certificates ensure domain owners know which onion addresses can be discovered for their sites, much like Certificate Transparency does the same thing for public TLS keys. This also adds censorship resistance to the discovery as logs are append-only.

As part of exploring Certificate Transparency in Tor Browser, we further contributed to the understanding of how the protocols used during website visits affect unlinkability between Tor users and their destination websites. For example, fetching an inclusion proof from a Certificate Transparency log is one such protocol. We extended the attacker model of website fingerprinting attacks with website oracles that reveal whether any network user visited a website during a specific time frame. Our results show that website oracles eliminate most false positives for all but the most frequently visited websites. In addition to the theoretical evaluation of the extended attacker model, we could exploit (timeless) timing attacks in Tor's DNS cache to instantiate real-world website oracles without any special capabilities or reach into third-parties. This led us to contribute to the understanding of how Tor's DNS cache performs today, including a proposal for a performant alternative that preloads the same popular domains on all Tor relays to withstand all (timeless) timing attacks.

As an outlook, our angle on Certificate Transparency verification has mostly been *reactive* for end-users. In other words, some or all certificate verification occurs asynchronously after a website visit. An alternative to this would be upfront delivery of inclusion proofs that reconstruct tree heads which witnesses cosigned; a form of *proactive* gossip as proposed by Syta *et al.* [103]. The significant upside is that the browser's verification could become non-interactive, eliminating privacy concerns and ensuring end-users only see certificates merged into the append-only logs. Investigating what the blockers for such a proposal are in practice—today—would be valuable as log verification quickly becomes complicated with signed certificate timestamps and reactive gossip-audit models. Are these blockers significant? Are they significant over time as other *eventual* changes will be needed, like post-quantum secure certificates? New transparency log applications are unlikely to need the complexity

of Certificate Transparency, and should likely not copy something that was designed to fit into an existing system with a large amount of legacy (such as certificate authorities, their established processes for certificate issuance, and the many client-server implementations already deployed on the Internet).

Orthogonal to the verification performed by end-users, contributing to the understanding of how domains (fail to) use Certificate Transparency for detecting mis-issued certificates is largely unexplored. For example, subscribing to email notifications of newly issued certificates becomes less useful in an era where certificates are renewed frequently and automatically. Instead, domain owners need easy-to-use solutions that raise alarms only if there is a problem.

Finally, the mitigation deployed to counter our (timeless) timing attacks in Tor's DNS cache is just that: a mitigation, not a defense, that applies to modestly popular websites but not the long tail where the base rate is low. This is because the attacker's needed website oracle time frame is so large that a fuzzy time-to-live value does nothing. Practical aspects of a preloaded DNS cache need to be explored further before deployment, such as the assumption of a third-party that visits popular domains to assemble an allowlist. We may also have *underestimated* the utility of the existing Umbrella list, which in and of itself does not require any new third-party. Does the use of Umbrella impact page-load latency? Latency is the most crucial parameter to keep minimized. The question is whether frequently looked-up domains are missed or not by skipping the website-visit step, as for the non-extended Alexa and Tranco lists.

More broadly, the question of how to strike a balance between *efficiency* and *effectiveness* of website fingerprinting defenses is open. How much overhead in terms of added latency and/or bandwidth is needed? How much of that overhead is sustainable, both from a user perspective (where, e.g., latency is crucial for web browsing and other interactive activities) and a network health perspective (such as the amount of volunteered relay bandwidth that is wasted)? It is paramount to neither overestimate nor underestimate attacker capabilities, which goes back to the still-debated threat model of website fingerprinting attacks. Regardless of if Tor's DNS cache becomes preloaded or not, it will be difficult to circumvent DNS lookups from happening. Someone—be it a weak attacker like ourselves or a recursive DNS resolver at an Internet service provider—is in a position to narrow down the destination anonymity set. This is especially true when also considering other protocols that reveal information about the destination anonymity set during website visits. Accepting that sources of real-world website oracles are prevalent implies that *the world can be closed*. Therefore, a closed world is more realistic than an open world.

Acknowledgments

I received valuable feedback while writing the introductory summary from Simone Fischer-Hübner Johan Garcia, Stefan Lindskog, and Tobias Pulls. The final draft was further improved with helpful nudges from Grammarly.

References

[1] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth D. Schoen, and Brad Warren. Let's Encrypt: An automated certificate authority to encrypt the entire web. In *CCS*, 2019.

[2] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE S&P*, 2013.

[3] Apple Inc. Apple's Certificate Transparency policy. https://support.apple.com/en-us/HT205280, accessed 2023-04-30.

[4] Andrew Ayer. Reliability of monitors | mitigations. https://groups.google.com/a/chromium.org/g/ct-policy/c/zCtQrn_7QK8, accessed 2023-04-30.

[5] Andrew Ayer. Retiring DigiCert log server (aka "CT1") in Chrome. https://groups.google.com/a/chromium.org/g/ct-policy/c/P5aj4JEBFPM/m/9AEcvY01EQAJ, accessed 2023-04-30.

[6] Andrew Ayer. Trust Asia 2021 has produced inconsistent STHs. https://groups.google.com/a/chromium.org/g/ct-policy/c/VJaSg717m9g, accessed 2023-04-30.

[7] Vaibhav Bajpai, Anna Brunström, Anja Feldmann, Wolfgang Kellerer, Aiko Pras, Henning Schulzrinne, Georgios Smaragdakis, Matthias Wählisch, and Klaus Wehrle. The dagstuhl beginners guide to reproducibility for experimental networking research. *CCR*, 49(1), 2019.

[8] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with BGP. In *USENIX Security*, 2018.

[9] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *ESORICS*, 2011.

[10] David Brumley and Dan Boneh. Remote timing attacks are practical. In *USENIX Security*, 2003.

[11] CA/Browser Forum. Baseline requirements for the issuance and management of publicly-trusted certificates. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.8.7.pdf, accessed 2023-04-30.

[12] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *ESORICS*, 2010.

[13] Melissa Chase and Sarah Meiklejohn. Transparency overlays and applications. In *CCS*, 2016.

[14] Heyning Cheng and Ron Avnur. Traffic analysis of SSL encrypted web browsing. *Project paper, University of Berkeley*, 1998.

[15] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world. In *USENIX Security*, 2022.

[16] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. In *CNS*, 2015.

[17] Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: revisiting past challenges and evaluating certificate trust model enhancements. In *IEEE S&P*, 2013.

[18] Sigsum Project Contributors. Witness API v0. https://git.glasklar.is/sigsum/project/documentation/-/blob/main/witness.md, accessed 2023-04-30.

[19] Scott A. Crosby and Dan S. Wallach. Efficient data structures for tamper-evident logging. In *USENIX Security*, 2009.

[20] Scott A. Crosby, Dan S. Wallach, and Rudolf H. Riedi. Opportunities and limits of remote timing attacks. *ACM Trans. Inf. Syst. Secur.*, 12(3), 2009.

[21] Rasmus Dahlberg. Transparency log preliminaries. https://gitlab.torproject.org/rgdd/ct/-/blob/main/doc/tlog-preliminaries.md, accessed 2023-04-30.

[22] Joe DeBlasio. Opt-out SCT auditing in Chrome. https://docs.google.com/document/d/16G-Q7iN3kB46GSW5b-sfH5MO3nKSYyEb77YsM7TMZGE/edit, accessed 2023-04-30.

[23] Peter J Denning. Is computer science science? *CACM*, 48(4), 2005.

[24] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[25] Alexandra Dirksen, David Klein, Robert Michael, Tilman Stehr, Konrad Rieck, and Martin Johns. LogPicker: Strengthening Certificate Transparency against covert adversaries. *PETS*, 2021(4).

[26] Gordana Dodig-Crnkovic. Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Skövde, Sweden*, 2002.

[27] Jason A. Donenfeld. Wireguard: Next generation kernel network tunnel. In *NDSS*, 2017.

[28] Benjamin Dowling, Felix Günther, Udyani Herath, and Douglas Stebila. Secure logging schemes and Certificate Transparency. In *ESORICS*, 2016.

[29] Graham Edgecombe. WoSign log failure to incorporate entry within the MMD. https://groups.google.com/a/chromium.org/g/ct-policy/c/-eV4Xe8toVk/m/pC5gSjJKCwAJ, accessed 2023-04-30.

[30] Adam Eijdenberg, Ben Laurie, and Al Cutter. Verifiable data structures. https://github.com/google/trillian/blob/master/docs/papers/VerifiableDataStructures.pdf, accessed 2023-04-30.

[31] Saba Eskandarian, Eran Messeri, Joseph Bonneau, and Dan Boneh. Certificate Transparency with privacy. *PETS*, 2017(4).

[32] Qian Ge, Yuval Yarom, David A. Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *JCEN*, 8(1), 2018.

[33] Google LLC. Certificate Transparency in Chrome. https://googlechrome.github.io/CertificateTransparency/ct_policy.html, accessed 2023-04-30.

[34] Google LLC. The list of existing monitors. https://certificate.transparency.dev/monitors/, accessed 2023-04-30.

[35] Benjamin Greschbach, Tobias Pulls, Laura M. Roberts, Phillip Winter, and Nick Feamster. The effect of DNS on Tor's anonymity. In *NDSS*, 2017.

[36] Lachlan J. Gunn, Andrew Allison, and Derek Abbott. Safety in numbers: Anonymization makes keyservers trustworthy. In *HotPETs*, 2017.

[37] Paul Hadfield. Google Aviator incident under investigation. https://groups.google.com/a/chromium.org/g/ct-policy/c/ZZf3iryLgCo/m/mi-4ViMiCAAJ, accessed 2023-04-30.

[38] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security*, 2023.

[39] Cormac Herley and Paul C. van Oorschot. SoK: Science, security and the elusive goal of security as a scientific pursuit. In *IEEE S&P*, 2017.

[40] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *CCSW*, 2009.

[41] Andrew Hintz. Fingerprinting websites using traffic analysis. In *PETS*, 2002.

[42] Benjamin Hof and Georg Carle. Software distribution transparency and auditability. *CoRR*, abs/1711.07278, 2017.

[43] Paul Hoffman and Patrick McManus. DNS queries over HTTPS (DoH). RFC 8484, IETF, 2018.

[44] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. Tracking the deployment of TLS 1.3 on the web: a story of experimentation and centralization. *CCR*, 50(3), 2020.

[45] Hans Hoogstraaten. Black tulip—report of the investigation into the DigiNotar certificate authority breach. Technical report, Fox-IT, 2012.

[46] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. Users get routed: traffic correlation on Tor by realistic adversaries. In *CCS*, 2013.

[47] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *CCS*, 2014.

[48] George Kadianakis, Yawning Angel, and David Goulet. A name system API for Tor onion services. https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/proposals/279-naming-layer-api.txt, accessed 2023-04-30.

[49] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. Revisiting user privacy for Certificate Transparency. In *IEEE EuroS&P*, 2019.

[50] Neal Koblitz and Alfred Menezes. Another look at "provable security". *J. Cryptol.*, 20(1), 2007.

[51] Neal Koblitz and Alfred Menezes. Another look at security definitions. *AMC*, 7(1), 2013.

[52] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, 1996.

[53] Ben Laurie. Certificate Transparency over DNS. https://github.com/google/certificate-transparency-rfcs/blob/master/dns/draft-ct-over-dns.md, accessed 2023-04-30.

[54] Ben Laurie. Certificate Transparency. *CACM*, 57(10), 2014.

[55] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, IETF, 2013.

[56] Ben Laurie, Eran Messeri, and Rob Stradling. Certificate Transparency version 2.0. RFC 9162, IETF, 2021.

[57] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. Certificate Transparency in the wild: Exploring the reliability of monitors. In *CCS*, 2019.

[58] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In *CCS*, 2006.

[59] Wouter Lueks and Ian Goldberg. Sublinear scaling for multi-client private information retrieval. In *FC*, 2015.

[60] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Oztürk, Kevin Lewi, and Sean F. Lawlor. Parakeet: Practical key transparency for end-to-end encrypted messaging. In *NDSS*, 2023.

[61] Akshaya Mani, T. Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. Understanding Tor usage with privacy-preserving measurement. In *IMC*, 2018.

[62] Macarena C. Martínez-Rodríguez, Ignacio M. Delgado-Lozano, and Billy Bob Brumley. SoK: Remote power analysis. In *ARES*, 2021.

[63] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. SoK: A critical evaluation of efficient website fingerprinting defenses. In *IEEE S&P*, 2023.

[64] Sarah Meiklejohn, Joe DeBlasio, Devon O'Brien, Chris Thompson, Kevin Yeo, and Emily Stark. SoK: SCT auditing in Certificate Transparency. *PETS*, 2022(3).

[65] Sarah Meiklejohn, Pavel Kalinnikov, Cindy S. Lin, Martin Hutchinson, Gary Belvin, Mariana Raykova, and Al Cutter. Think global, act local: Gossip and client audits in verifiable data structures. *CoRR*, abs/2011.04551, 2020.

[66] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: bringing key transparency to end users. In *USENIX Security*, 2015.

[67] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, 1987.

[68] Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *CCS*, 2011.

[69] Alec Muffett. Real-world onion sites. https://github.com/alecmuffett/real-world-onion-sites, accessed 2023-04-30.

[70] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *IEEE S&P*, 2005.

[71] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. DeepCorr: Strong flow correlation attacks on Tor using deep learning. In *CCS*, 2018.

[72] Linus Nordberg, Daniel Kahn Gillmor, and Tom Ritter. Gossiping in CT. Internet-draft draft-ietf-trans-gossip-05, IETF, 2018.

[73] Juha Nurmi. *Understanding the Usage of Anonymous Onion Services*. PhD thesis, Tampere University, Finland, 2019.

[74] Se Eun Oh, Taiji Yang, Nate Mathews, James K. Holland, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. DeepCoFFEA: Improved flow correlation attacks on Tor via metric learning and amplification. In *IEEE S&P*, 2022.

[75] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *WPES*, 2011.

[76] Mike Perry. A critique of website traffic fingerprinting attacks. https://blog.torproject.org/critique-website-traffic-fingerprinting-attacks, accessed 2023-04-30.

[77] Mike Perry, Erinn Clark, Steven Murdoch, and Georg Koppen. The design and implementation of the Tor Browser [DRAFT]. https://2019.www.torproject.org/projects/torbrowser/design/, accessed 2023-04-30.

[78] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2010.

[79] Tor Project. Browse privately. Explore freely. Defend yourself against tracking and surveillance. Circumvent censorship. https://www.torproject.org/, accessed 2022-04-30.

[80] Tor Project. Onion-Location. https://community.torproject.org/onion-services/advanced/onion-location/, accessed 2023-04-30.

[81] Tor Project. Research safety board. https://research.torproject.org/safetyboard/, accessed 2023-04-30.

[82] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. Tik-Tok: The utility of packet timing in website fingerprinting attacks. *PETS*, 2020(3).

[83] Vera Rimmer, Theodor Schnitzler, Tom van Goethem, Abel Rodríguez Romero, Wouter Joosen, and Katharina Kohls. Trace oddity: Methodologies for data-driven traffic analysis on Tor. *PETS*, 2022(3).

[84] Jeremy Rowley. CT2 log compromised via Salt vulnerability. https://groups.google.com/a/chromium.org/forum/#!topic/ct-policy/aKNbZuJzwfM, accessed 2023-04-30.

[85] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 Internet public key infrastructure online certificate status protocol—OCSP. RFC 6960, IETF, 2013.

[86] Sectigo Limited. crt.sh: certificate search. https://github.com/crtsh, accessed 2023-04-30.

[87] Sectigo Limited. crt.sh: certificate search ID = '8913351873'. https://crt.sh/?id=8913351873, accessed 2023-04-30.

[88] SecureDrop. Getting an onion name for your SecureDrop. https://securedrop.org/faq/getting-onion-name-your-securedrop/, accessed 2023-04-30.

[89] Sandra Siby, Marc Juárez, Claudia Díaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS -> privacy? A traffic analysis perspective. In *NDSS*, 2020.

[90] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *CCS*, 2018.

[91] Ryan Sleevi. StartCom log misbehaving: Failure to incorporate SCTs. https://groups.google.com/a/chromium.org/g/ct-policy/c/92HIh2vG6GA/m/hBEHxcpoCgAJ, accessed 2023-04-30.

[92] Ryan Sleevi. Upcoming CT log removal: Izenpe. https://groups.google.com/a/chromium.org/forum/#!topic/ct-policy/qOorKuhL1vA, accessed 2023-04-30.

[93] Ryan Sleevi. Upcoming log removal: Venafi CT log server. https://groups.google.com/a/chromium.org/forum/#!topic/ct-policy/KMAcNT3asTQ, accessed 2023-04-30.

[94] Ryan Sleevi and Eran Messeri. Certificate Transparency in Chrome: Monitoring CT logs consistency. https://docs.google.com/document/d/1FP5J5Sfsg0OR9P4YT0q1dM02iavhi8ix1mZlZe_z-ls/edit?pref=2&pli=1, accessed 2023-04-30.

[95] SSLMate Inc. Cert spotter—Certificate Transparency monitor. https://github.com/SSLMate/certspotter, accessed 2023-04-30.

[96] SSLMate Inc. Timeline of certificate authority failures. https://sslmate.com/resources/certificate_authority_failures, accessed 2023-04-30.

[97] Emily Stark, Joe DeBlasio, Devon O'Brien, Davide Balzarotti, William Enck, Samuel King, and Angelos Stavrou. Certificate Transparency in Google Chrome: Past, present, and future. *IEEE S&P*, 19(6), 2021.

[98] Emily Stark, Ryan Sleevi, Rijad Muminovic, Devon O'Brien, Eran Messeri, Adrienne Porter Felt, Brendan McMillion, and Parisa Tabriz. Does Certificate Transparency break the web? Measuring adoption and error rate. In *IEEE S&P*, 2019.

[99] Emily Stark and Chris Thompson. Opt-in SCT auditing. https://docs.google.com/document/d/1G1Jy8LJgSqJ-B673GnTYIG4b7XRw2ZLtvvSlrqFcl4A/edit, accessed 2023-04-30.

[100] Nick Sullivan. Understanding use-cases for SCTs delivered via OCSP stapling for TLS extension. https://groups.google.com/a/chromium.org/g/ct-policy/c/WX6iZt7uJBs, accessed 2023-04-30.

[101] Nick Sullivan and Sean Turner. Messaging layer security: Secure and usable end-to-end encryption. https://www.ietf.org/blog/mls-secure-and-usable-end-to-end-encryption/, accessed 2023-04-30.

[102] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE S&P*, 2002.

[103] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *IEEE S&P*, 2016.

[104] Alin Tomescu, Vivek Bhupatiraju, Dimitrios Papadopoulos, Charalampos Papamanthou, Nikos Triandopoulos, and Srinivas Devadas. Transparency logs via append-only authenticated dictionaries. In *CCS*, 2019.

[105] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES implemented on computers with cache. In *CHES*, 2003.

[106] Tom van Goethem, Christina Pöpper, Wouter Joosen, and Mathy Vanhoef. Timeless timing attacks: Exploiting concurrency to leak secrets over remote connections. In *USENIX Security*, 2020.

[107] Mathy Vanhoef and Tom Van Goethem. HEIST: HTTP encrypted information can be stolen through TCP-windows. In *Black Hat US Briefings*, 2016.

[108] Jesse Victors, Ming Li, and Xinwen Fu. The onion name system. *PETS*, 2017(1).

[109] Emanuel von Zezschwitz, Serena Chen, and Emily Stark. "It builds trust with the customers"—exploring user perceptions of the padlock icon in browser UI. In *IEEE SPW*, 2022.

[110] Jun Wang, Weinan Zhang, and Shuai Yuan. Display advertising with real-time bidding (RTB) and behavioural targeting. *Foundations and Trends in Information Retrieval*, 2017.

[111] Tao Wang and Ian Goldberg. On realistically attacking Tor with website fingerprinting. *PETS*, 2016(4).

[112] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In *USENIX Security*, 2022.

[113] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar R. Weippl. Spoiled onions: Exposing malicious Tor exit relays. In *PETS*, 2014.

[114] Zerodium. We pay big bounties. https://zerodium.com/, accessed 2023-04-30.

On Certificate Transparency Verification and Unlinkability of Websites Visited by Tor Users

Certificate Transparency is an ecosystem of logs, monitors, and auditors that hold certificate authorities accountable while issuing certificates. We show how the amount of trust that TLS clients and domain owners need to place in Certificate Transparency can be reduced, both in the context of existing gradual deployments and the largely unexplored area of Tor. Our contributions include improved third-party monitoring, a gossip protocol plugging into Certificate Transparency over DNS, an incrementally deployable gossip-audit model tailored for Tor Browser, and using certificates with onion addresses. The methods used range from proof sketches to Internet measurements and prototype evaluations. An essential part of our evaluation in Tor is to assess how the protocols used during website visits–such as requesting an inclusion proof from a Certificate Transparency log–affect unlinkability between senders and receivers. We find that most false positives in website fingerprinting attacks can be eliminated for all but the most frequently visited sites. This is because the destination anonymity set can be reduced due to how Internet protocols work: communication is observable and often involves third-party interactions. Some of the used protocols can further be subject to side-channel analysis. For example, we show that remote (timeless) timing attacks against Tor's DNS cache reliably reveal the timing of past exit traffic. The severity and practicality of our extension to website fingerprinting pose threats to the anonymity provided by Tor. We conclude that access to a so-called website oracle should be an assumed attacker capability when evaluating website fingerprinting defenses.

ISBN 978-91-7867-372-8 (print)

ISBN 978-91-7867-373-5 (pdf)

ISSN 1403-8099

DOCTORAL THESIS | Karlstad University Studies | 2023:15