Erik Sy*, Christian Burkert, Hannes Federrath, and Mathias Fischer

A QUIC Look at Web Tracking

Abstract: QUIC has been developed by Google to improve the transport performance of HTTPS traffic. It currently accounts for approx. 7% of the global Internet traffic. In this work, we investigate the feasibility of user tracking via QUIC from the perspective of an online service. Our analysis reveals that the protocol design contains violations of privacy best practices through which a tracker can passively and uniquely identify clients across several connections. This tracking mechanisms can achieve reduced delays and bandwidth requirements compared to conventional browser fingerprinting or HTTP cookies. This allows them to be applied in resource- or time-constrained scenarios such as real-time biddings in online advertising. To validate this finding, we investigated browsers which enable QUIC by default, e.g., Google Chrome. Our results suggest that the analyzed browsers do not provide protective measures against tracking via QUIC. However, the introduced mechanisms reset during a browser restart, which clears the cached connection data and thus limits achievable tracking periods. To mitigate the identified privacy issues, we propose changes to QUIC's protocol design, the operation of QUIC-enabled web servers, and browser implementations.

Keywords: QUIC Transport Protocol, Network Security, Protocol Design, Privacy Protections, Browser Measurements

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

The QUIC protocol is designed with the aim to provide privacy assurances comparable to TLS [35]. To achieve this goal, QUIC traffic is mostly encrypted with the exception of a few early handshake messages. Furthermore, QUIC switches potential client identifiers such as the *source-address token* frequently to limit the possibility for network-based attackers, e.g., internet service providers or intelligence services, to track users across several connections.

In the light of mass surveillance, network-based attackers represent a legitimate concern. However, online tracking for advertising or web analytics poses a similar threat to users' privacy. Therefore, browsers are required to protect their users' privacy against tracking through web servers that is not conducted in mutual agreement. Nonetheless, there is a trade-off between performance and privacy for browsers, in which reduced user privacy can yield a higher browser performance. For example, the usage of TLS session resumption mechanisms accelerates connection establishment, but can be used for user tracking at the same time [34]. Browsers such as Google Chrome or Opera balance this privacy versus performance trade-off by limiting the time frame of TLS session resumption [34]. QUIC allows to establish secure connections with a zero round-trip time (0-RTT) handshake. The widespread TCP/TLS 1.2 alternative requires at least three round-trips [18].

In this work we investigate the impact of QUIC's performance enhancements on user privacy. We find that QUIC's *source-address token* and QUIC's *server config* can be used by a server to identify a user across several connections. These tracking approaches exploit that a client stores data from the server for reuse in a subsequent connection. Both mechanisms allow a user identification based on the first message that a client sends to establish a connection with 0-RTT.

Compared to common online tracking practices such as HTTP cookies [10] or browser fingerprinting [1], the presented tracking mechanisms can provide performance advantages. They do not require an tracker to actively request information from a user's browser. This saves additional delays and bandwidth consumption, which otherwise restricts the practical applicability of such tracking methods [2]. For example, a third-party tracker hosting popular web fonts would directly affect the page load time of a website by performing browser fingerprinting and consequently impair the user experience [15]. Furthermore, the additional delay is a disadvantage for the highly time-constrained practice of

***Corresponding Author: Erik Sy:** University of Hamburg, E-mail: sy@informatik.uni-hamburg.de
Christian Burkert: University of Hamburg, E-mail: burkert@informatik.uni-hamburg.de
Hannes Federrath: University of Hamburg, E-mail: federrath@informatik.uni-hamburg.de
Mathias Fischer: University of Hamburg, E-mail: mfischer@informatik.uni-hamburg.de

real-time bidding in online advertising [22]. QUIC-based tracking does not come with these drawbacks.

To the best of our knowledge, we are the first to report on user tracking via the QUIC protocol. The main contributions of our paper are:

- We describe tracking via QUIC's *source-address token*, which allows online services to link several website visits to the same user. Furthermore, we present tracking via QUIC's *server config*, which enables online services and also network-based attackers to track users across multiple sessions.
- We investigate the configuration of browsers that support QUIC connections by default. Our results indicate that the investigated browser configurations do not restrict the presented tracking mechanism. Furthermore, the tested browsers do not prevent third-parties from exploiting the presented mechanisms to track users across different visited websites.
- We propose countermeasures that mitigate the presented tracking mechanisms. We address tracking via the *server config* by ensuring that users attempt 0-RTT handshakes only if they observe the same *server config* during a website visit. To mitigate tracking via the *source-address token* we provide an action list for browser vendors to improve user privacy. Subsequently, we review an alternative design for QUIC's connection establishment aiming for better privacy protections.

The remainder of this paper is structured as follows: Section 2 describes the background on the QUIC protocol handshake. Section 3 reviews QUIC's tracking mechanisms and their resulting privacy problems. Section 4 summarizes our major results on tracking via the QUIC protocol, before we discuss the practical impact of our results and possible countermeasures in Section 5. Related work is reviewed in Section 6, and Section 7 concludes the paper.

2 Background on QUIC's handshake

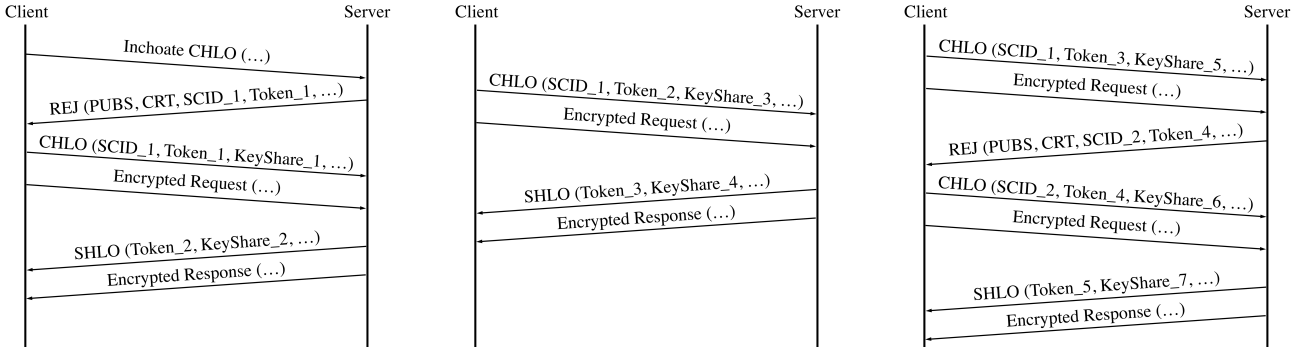
To set up a secure transport connection, the QUIC protocol incorporates some functionality of TCP, TLS, and HTTP/2. QUIC's design allows to concurrently conduct the cryptographic and the transport handshake, which provides performance improvements. In this section, we describe the cryptographic handshake of the QUIC protocol.

QUIC provides two modes for the connection setup: The zero round-trip time (0-RTT) and the one round-trip time (1-RTT) full handshake, as shown in Figure 1. The full handshake is required for the initial connection between the client and the server. Subsequent handshakes can use the abbreviated 0-RTT mode utilizing cached information from previous connections between the respective client-server pair. This behavior is indicated by reusing the *server config identifier* `SCID_1` and *source-address token* `Token_2` from the connection shown in Figure 1 a) in the subsequent handshake (see Figure 1 b). The SCID describes a 16-byte identifier that allows the peers to reference a specific *server config*, whereas the token describes an encrypted and authenticated block which is opaque to the client. The token contains the client's publicly visible IP address and a timestamp as seen by the server.

The initial handshake starts with the client sending an *inchoate client hello* (CHLO) message, as shown in Figure 1 a). This message contains a randomly generated *connection identifier* (ConnID), which is used by both parties to refer to this connection. The ConnID is an optional part of the public header of a QUIC packet. Furthermore, the ConnID allows migrating a connection to an endpoint's new IP address and/or port. This becomes necessary after NAT rebinding [31] or when a new IP address gets assigned to an endpoint.

Next, the server responds with a *reject* (REJ) message. This message contains among others: (i) the *server config* including the server's long-term Diffie-Hellmann public value (PUBS), (ii) a certificate chain authenticating the server (CRT), (iii) a signature from the server's private key, (iv) a *server config identifier* (SCID) and (v) a *source-address token* (token). While (i)-(iii) are required to authenticate the server's identity and to establish the secure channel, the token is used to authenticate the client's publicly visible IP address. Subsequently, the client responds with a complete CHLO. Now, the client can compute the *initial keys* as a shared value of the server's PUBS and its own ephemeral Diffie-Hellman private key. The *initial keys* can be used to send encrypted requests to the server before the server responds to the complete CHLO. Note, that data encrypted with the initial keys do not provide forward-secrecy [17].

The server can compute the *final* and forward-secure key based on the client's *ephemeral Diffie-Hellman public value* (KeyShare) contained in the complete CHLO. Therefore, the *server hello* (SHLO) and all consecutive messages are forward-securely encrypted.



a) 1-RTT full handshake b) 0-RTT handshake c) Rejected 0-RTT handshake

Fig. 1. Handshakes in QUIC transport protocol.

The SHLO message contains amongst others a new token, the server's KeyShare, and the ConnID.

For a 0-RTT handshake as shown in Figure 1 b), the client follows the same protocol starting from the complete CHLO. Thus, the client needs to cache the *server config* received with the REJ message and the latest received source-address token for subsequent 0-RTT connection setups.

Moreover, the server can reject the client's 0-RTT handshake attempt as shown in Figure 1 c). Failures like an invalid token or an expired server config presented by the client can cause the server to reject. If this occurs, the server will reply with a REJ message and both parties can proceed with the initial handshake protocol starting from its third message.

3 Tracking via QUIC

QUIC's novel approach to build secure 0-RTT connections uses long-term Diffie-Hellmann public values for the key exchange. This novelty as well as the mixing of cryptographic and transport-layer handshakes, make it advisable to conduct a privacy analysis of this protocol design.

In this section, we analyze which parameters of QUIC's handshake can be used by a malicious server to track a user across several connections. Subsequently, we describe the impact of QUIC's tracking duration on user privacy. Subsequently, we review the consequences of the unrestricted use of QUIC's cross-session identifiers by a third-party online tracker.

3.1 QUIC's identifiers suitable for user tracking

The QUIC protocol includes identifiers which are bound to a single user session such as the ConnID. However, to track users' browsing behavior over long periods, a tracker needs to link consecutive user sessions to each other. Thus, the tracker requires an identifier, which allows recognizing the user across multiple sessions.

A QUIC protocol client is required to cache the retrieved *server config* and the *source-address token* across sessions and to present them to the server when establishing a new 0-RTT connection. Therefore, it needs to be investigated, whether this cached data can be exploited by a server as a storage-based tracking mechanism that assigns unique identifiers to users.

We observe, that the *server config* contains several tag/value pairs that may be used as a hidden channel for user tracking. For example, the server can assign a distinct *server config identifier* (SCID) to each user, which provides 16-byte of entropy. A tracking server would then be able to link the connection in which a unique *server config* was initially issued to those connections where the client provides the same *server config* within its CHLO.

Note, that the tag/value pairs of the *server config* such as the SCID are transmitted in cleartext within the CHLO and the REJ message. Thus, also a passive observer of the exchanged messages can identify a user based on the used unique SCID.

The *source-address token* is a unique, authenticated-encrypted data block provided by the server, which cannot be decrypted by the client. For the purpose of IP address spoofing prevention, it contains the user's publicly visible IP address and a timestamp as seen by the server. The client caches the *source-*

address token and presents it to the server during the setup of a new connection. This allows a server to link the connection where the *source-address token* is initially issued with each subsequent connections where the same token is presented during the CHLO message. Finally, this enables the server to identify a chain of connections associated with a user.

Moreover, a rejected 0-RTT handshake as shown in Figure 1 c) does not terminate a tracking period, because the server can link the observed *source-address token* and/or *server config* from the inchoate CHLO with the newly assigned ones in the server's REJ message.

3.2 Achievable tracking periods

Always on and *always with* are characteristics of mobile devices such as smartphones and tablets that provide ubiquitous access to the Internet. These mobile devices account already for about half of all web browsing activity [32]. On such devices, a web browser along with its cache can remain active for several days in the background of the operating system. Thus, also the retrieved *server configs* and tokens remain accessible until the web browser is restarted or they eventually expire.

To protect users against tracking via QUIC, it is required to restrict the maximum tracking duration independently of browser restarts. The QUIC protocol provides the *server config time to live* (STTL), which describes a hint by the server about the remaining lifetime of the provided *server config*. The STTL could constitute an upper limit to feasible tracking periods via QUIC's *server config*. This would be the case, if the client no longer used previously retrieved *server configs* or *source-address tokens* to establish connections after the STTL expired.

The protocol design must not allow a prolongation of the STTL if it shall serve as an effective upper limit for tracking periods. However, the documentation of the QUIC protocol [36] describes that server config update (SCUP) messages are used to refresh the *server config* and *source-address token*. Thus, SCUP messages can extend the period over which 0-RTT connections can be established by overwriting the cached data with the fresh *server config* and its corresponding STTL. This leads to a situation, in which every server that is revisited within the given STTL is able to extend the tracking period beyond the STTL by making use of SCUP messages. Note that the documentation of QUIC [12, 35] does not describe a maximum tracking duration regarding prolongations based on SCUP messages.

Moreover, the QUIC protocol does not include a mechanism that explicitly restricts tracking periods based on QUIC's *source-address token*. Thus, it depends on the QUIC implementation to restrict tracking via *source-address tokens*.

3.3 Third-party tracking

To collect comprehensive profiles about a user's browsing behavior, third-party tracking can be applied. Third-party tracking refers to a practice, where a party, other than the targeted website, can track a user's visit. It is a widespread phenomenon on the Internet with an average of 17.7 third-party trackers per website across the Alexa Top 500 categories [9]. Especially Google with its various hostnames is present on nearly 80% of the Alexa Top Million Sites [3, 9] and thus can gain deep insights into users' browsing behavior.

As for tracking via QUIC, third-party trackers can recognize users based on the token that the latter present during the 0-RTT connection attempt. Thus, a tracker can link multiple observed visits of a user across sites, where the tracker is present as a third-party. However, to distinguish the various first-party sites a user visited, the tracker requires an identifier such as an HTTP referrer or a custom URL per first-party.

Note, that the documentation of QUIC [12, 35] does not contain information on the protection of users against such third-party tracking.

4 Evaluation

To evaluate the feasibility of user tracking via the QUIC protocol, we investigate the default configuration of browsers supporting the QUIC protocol. Subsequently, we analyze the configuration of QUIC servers deployed by popular websites.

4.1 Evaluation of browser configurations

The default configuration of browsers can significantly contribute to the protection of users' privacy against online trackers. In this section, we present our measurement of browsers' QUIC configuration.

We sampled the most popular browsers as provided by [25] to investigate if they enable the QUIC protocol by default. Furthermore, we included the Chromium browser to our set of test candidates, because this rela-

tively popular browser employs the same user-agent as the Chrome browser and would thus not be listed separately. We found that the QUIC protocol is supported by Chrome, Chromium, and Opera on desktop operating systems and by Chrome on the Android mobile operating system.

To assess the default configuration of these browsers, we intercepted and analyzed the network traffic between the browser and a provided website that supports the QUIC protocol. We captured the network traffic using Wireshark [7] which ran on the same computer as the tested browser. For the Android device running the mobile version of Chrome, we provided Internet access over a dedicated Wi-Fi, where we intercepted the network traffic on the Wi-Fi access point. To investigate the encrypted network traffic, we used the network logging systems of these browsers [28].

During our measurements, the browsers remained running in the background of their operating systems and we explicitly did not restart or update the browsers during our measurements.

4.1.1 Token lifetime

With our first test, we investigate the time until a browser no longer uses a previously assigned token when revisiting a website. We refer to this period as the token lifetime. To assess the token lifetime, we visit a website and validate that the connection was established with the QUIC protocol. Afterwards, we close the browser tab which we used to visit the website and leave the browser idle in the background of the operating system for a given period of time. Next, we revisit the website with the browser and analyze the network traffic to observe whether the browser transmitted an old token during the handshake. We repeated this measurement for increasingly longer periods of intermediate waiting.

Table 1. Browsers' default QUIC configuration

Plt.	Browser	Lower boundary of token lifetime	Honors STTL	Third-party tracking
Desktop	Chrome	11 days	no	viable
	Opera	11 days	no	viable
	Chromium	11 days	no	viable
Android	Chrome	11 days	no	viable

Our results suggest that the token lifetime is unrestricted in the evaluated browsers (see Table 1). However, on the basis of our measurements, we can only exclude the existence of such restrictions for the first 11 days for the investigated browsers. This duration substantiates that the investigated browsers do not restrict this tracking mechanism.

4.1.2 STTL adherence

Our second test examines whether browsers transmit the cached server config and the cached token after the *server config time to live* has expired. We measured this browser behavior by visiting a website that provided a *server config* with a STTL of two days. Following this, we closed all browser tabs with a connection to that site and the browser remained idle in the background of the operating system. After the STTL expired, we reconnected to the website and observed whether the browser still transmitted the expired *server config* and token during the first reconnection attempt.

As shown in Table 1, none of the investigated browsers stopped using the *server config* after the STTL expired. Furthermore, we found that all tested browsers continued to send their cached *source-address token* in the subsequent CHLO after the STTL expired. Therefore, the observed browser configurations indicate that not even STTLs as short as the duration of a single page visit would protect against the presented tracking mechanisms, if browsers do not enforce the STTL expiry themselves.

4.1.3 Third-party tracking

In the third browser measurement, we assess whether third-party tracking is feasible via the browser's QUIC configuration. For this purpose, we load a website $\mathcal{A}$ which includes a third-party $\mathcal{T}$, as it is shown in Figure 2. Afterwards, we close all browser tabs and wait for all QUIC connections to time out. The upper limit for an idle timeout is specified in the QUIC protocol as ten minutes for implicit connection shutdowns [36]. Next, we retrieve website $\mathcal{B}$ which includes the same third-party $\mathcal{T}$. Based on the intercepted network traffic, we analyze this second QUIC handshake with $\mathcal{T}$, which we conducted within the context of the website $\mathcal{B}$. If a token or *server config* from $\mathcal{A}$'s context has been reused to connect to $\mathcal{T}$ in the context of visiting $\mathcal{B}$, then we con-

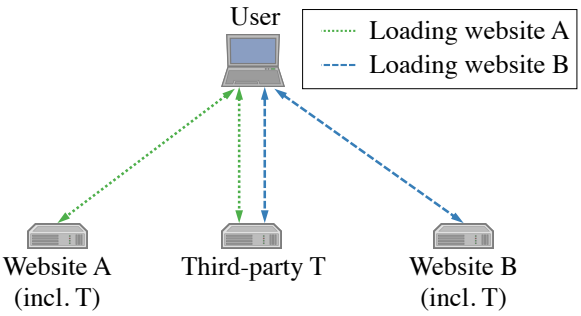


Fig. 2. Testbed to measure browser behavior in regard to third-party tracking.

clude that third-party tracking is feasible for the tested browser.

Our results indicate that all investigated browsers do not protect against third-party tracking via QUIC (see Table 1). This highlights the practical impact of the presented tracking approach, because third-party tracking is a wide-spread practice on the web [30].

4.2 Evaluation of server configurations

In the server-part of our evaluation, we analyse the degree of distribution of QUIC-enabled web sites, the configuration of QUIC's *server config time to live* among these sites, and their turnover of *server configs*.

4.2.1 Availability of QUIC-enabled servers

In our next experiment, we investigate the deployment of QUIC-enabled servers within the Alexa Top Million websites [3]. This provides an approximation of the total share of QUIC-enabled websites on the Internet.

We conducted this measurement on the 18th of June 2018 from the campus of the University of Hamburg. To scan the Alexa Top Million Sites, we applied a similar methodology as used in prior research work [29]. Thus, we used the tool Zmap [8] with the gQUIC-module [29] to send a *client hello* message to UDP port 443 of each Alexa-listed host. If the server responded with a valid *reject* message, we concluded that the server is supporting the QUIC protocol.

We found 186 websites within the Alexa Top Million that support the QUIC protocol. Of these sites, 141 were related to Google. The remaining 45 sites such as www.paris.fr, www.seagate.com or www.sony.jp employ a similar configuration, but we could not identify a common entity behind these websites. Table 2 presents the share of QUIC-enabled websites within different Alexa Top lists. With 20% and 21%, we find a significant adoption of QUIC within the Top Ten and the Top Hundred Sites, respectively. However, this share decreases for larger Top Alexa lists to only 0.0186% within the Alexa Top 1 Million. As several top ranked websites deploy the QUIC protocol, the share of the QUIC protocol on the global Internet traffic accounts for approximately seven percent [18].

Table 2. Websites with QUIC support in Alexa Top lists

Alexa Top list	Share with QUIC support
Alexa Top 10	20.00%
Alexa Top 100	21.00%
Alexa Top 1K	8.10%
Alexa Top 10K	1.69%
Alexa Top 100K	0.19%
Alexa Top 1M	0.02%

The small number of websites supporting QUIC indicates that the number of online trackers potentially exploiting the presented mechanisms is still small. However, the impact of a tracking mechanism is also significantly influenced by the size of the affected user base and the length of feasible tracking periods. In 2019, around 60% of global Internet users are affected by the presented tracking mechanism, as this is the market share of the Chrome browser [33]. Thus, any entity aiming to track its users, can deploy QUIC on its servers and achieve tracking periods of longer than 10 days for a significant share of its users (see Section 4.1).

Our measurement investigates still an early stage of QUIC's deployment on the Internet. For example, the upcoming third version of HTTP will use QUIC instead of TCP as the transport protocol [4]. Thus, we expect that more browser vendors and websites will add support for QUIC to their products and services within the next years.

4.2.2 Server config time to live

To analyze the default configuration of the identified 186 QUIC-enabled websites, we used the tool quic-grabber [29] to establish QUIC connections to them. We captured the network traffic of the respective handshakes with Wireshark [7] and extracted the handshake parameters. We investigated the *server config time to live* (STTL) issued by the server, because it may be

used by a QUIC client to temporally limit the usage of the presented tracking mechanisms.

We observed that servers issue the same *server config* with a decreasing STTL over time, until the server starts distributing a fresh *server config*. To account for this server behavior, we collected the *server configs* of these servers within intervals of ten minutes for a two-day period.

Our results indicate, that the domains belonging to Google deploy STTLs between 32 and 48 hours. Divergently, the other 45 domains exhibited STTLs between 4311 and 4320 hours, which are approximately 180 days. Note, that an STTL of 180 days as configured by these web servers does not contribute to the protection of user privacy, because it allows online trackers to aggregate long-term user profiles containing characteristic browsing behavior.

STTLs of 180 days indicate that a server allows for uninterrupted 0-RTT connection establishment within this period, provided that clients use the respective *server config*. Consequently, the server is required to securely store and hold the private keys corresponding to the Diffie-Hellmann public values (PUBS) that were issued as part of the respective *server config* for at least this period. Thus, from a security perspective, it seems reasonable to considerably reduce the observed STTLs of 180 days.

4.2.3 Server config turnover

Our measurements also indicate that the investigated web servers issue new *server configs* at least every twelve hours. This enables a server to refresh a client's *server config* at similar intervals upon a client's revisit by using SCUP messages (see Section 3.2). If we assume, that servers update *server config* whenever possible, then only a small fraction of website revisits would benefit from such long STTLs as 180 days compared to a two day STTL, because empirically, 87.3% of all website revisits occur within the first two days after the previous visit [34].

5 Discussion

In this section, we discuss the practical impact of the presented tracking mechanisms as well as protective measures against tracking via QUIC.

5.1 Practical impact of tracking via QUIC

To discuss its practicality, we compare tracking via QUIC to conventional browser fingerprinting, HTTP cookies and IP address-based tracking. We present use-cases in which the presented tracking approach is advantageous compared to these other popular tracking mechanisms, and discuss how the limitation of tracking interruptions through browser restarts can be overcome by using a secondary tracking mechanism as a fallback.

5.1.1 Comparison to other tracking mechanisms

As shown in Table 3, every considered tracking mechanism has its limitations, which reduces its practical applicability. Consequently, the highest long-term user identification rates can be achieved by combining different tracking mechanisms which compensate individual limitations. However, the simultaneous execution of multiple tracking mechanisms would increase client- and server-side resource consumption, both computationally and bandwidth-wise. To save resources, it seems interesting to an online tracker to prioritize tracking via QUIC over browser fingerprinting and HTTP cookies. Tracking via QUIC achieves unique user identification and is completed with the receipt of the CHLO message. It reduces the bandwidth overhead and tracking delay compared to browser fingerprinting. In contrast, browser fingerprinting negatively impacts the page load time of a website and thus might not be applicable in contexts where, for example, a third-party tracker hosts web fonts required for the rendering of the website. Tracking with QUIC can be applied even under such constrained circumstances.

Compared to HTTP cookies, QUIC-based tracking reduces the tracking delay in two scenarios: Either the encrypted request that includes the client's HTTP cookie is sent over a forward-secure connection or the server rejects the encrypted client request. In both scenarios, the server can track the user via QUIC one round trip before the server can decrypt the client's request to obtain the included HTTP cookie. This makes user identification via QUIC favorable in highly time-constrained scenarios such as real-time biddings for online advertising [37].

Just like QUIC-based tracking, IP address-based tracking is fully passive and does not introduce additional computational or communication overhead. However, IPv4 addresses are often shared nowadays [5],

Table 3. Comparison of tracking via QUIC to widely used tracking mechanisms.

Properties	Browser Fingerprinting	HTTP Cookies	IP Address Tracking	Tracking via QUIC
Tracking delay	additional round-trip time after connection setup & processing time	requires completed connection setup	none	none
Bandwidth requirements	additional requests & response	low	low	low
Accuracy	limited [19]	unique	limited	unique
Coverage	HTTP connections (restrictions apply [24])	HTTP connections (restrictions apply [23])	IP connections	QUIC connections
Survives a change of IP Address	yes	yes	no	yes
Survives a browser restart	yes	yes (restrictions apply [14])	yes	no

which renders this approach less accurate than tracking via QUIC with respect to unique user identification.

5.1.2 Limitation of QUIC-based tracking

Continuous tracking via QUIC is only possible as long as the browser is not restarted, because this clears the cached state of prior QUIC connections. We note that mobile devices are *always on* and rarely restarted. Furthermore, desktop operating systems provide a *sleep* mode which presumably reduces the occasions to restart the OS. Thus, the OS provides only limited restrictions to feasible tracking periods. Moreover, the avoidance of browser restarts allows the user to easily continue a prior browsing session as respective browser tabs are still available. Thus, we assume that incentives exist for the user to seldom restart a browser. From our perspective, tracking periods of several days or even weeks are feasible under real-world circumstances for a significant part of users.

The QUIC-based tracking mechanism is further limited by the ability of active network-based attackers to alter the *server config* and *source-address token*. These identifiers are transmitted in cleartext over the network within the CHLO and REJ messages. In the following, we describe the possible impact of such malicious behavior on user identification.

The server's REJ message and the client's CHLO message allow a network-based attacker to retrieve the *source-address token* and the *server config* which the server assigned to a user. Subsequently, the network-based attacker can establish own connections to the server, which make use of the observed *source-address token* and *server config*. Thereby, the network-based attacker would impersonate the user from the perspective of a tracking server.

Furthermore, the attacker can manipulate a server's REJ message and insert a different *source-address token* and/or *server config*. This causes the client to use this inserted data during the subsequent connection setup. However, a connection between the client-server pair can only be successfully established if the server is able to cryptographically validate the provided *source-address token* and *server config*. In exploiting this behavior, an attacker can trick a user to impersonate another user by inserting the latter's *source-address token* and/or *server config* into the former's server response.

As investigated by Lychev et al. [21], the manipulation of the user's *source-address token* or *server config* within the CHLO message will lead to failures within the connection establishment, because they are used as input into the encryption key derivation process. Therefore, the client and server will compute different encryption keys for the connection in such an event, which leads to a communication failure.

As a consequence, a network-based attacker can—to a limited extent—manipulate the *source-address token* and the *server config* that a client uses for the subsequent connection establishment. Hence, a user identification based on the presented tracking mechanisms can be misled by a network-based attacker.

5.2 Countermeasures

A simple and effective countermeasure against user tracking via the QUIC protocol is to establish every new connection with a full handshake. However, this prevents 0-RTT connection establishment and thus leads

to performance losses during website revisits. In the following, Section 5.2.1 presents measures to protect users against tracking via the *server config*. To mitigate the privacy issues originating from *source-address tokens*, we propose directly applicable measures for browser vendors in Section 5.2.2. Subsequently, we review the design of a privacy-friendly connection establishment in Section 5.2.3.

5.2.1 Protection against tracking via the *server config*

This privacy problem is inherent to QUIC's handshake design which relies on previously shared PUBS for connection establishment. To solve this problem, we propose a mechanism to monitor issued *server configs*. Note, that this proposal is similar to the established certificate transparency [20] where issued TLS certificates are collected in logs. To gather data, every user can submit an observed *server config* to such a log. Alternatively, the server operator can submit the issued *server config* to a log and provide the user with a cryptographic signature from the log operator to proof this transaction. Note, that the latter approach is similar to the *Signed Certificate Timestamp* [20] extension of TLS. Such a log allows to discover server operators that issue a large number of unique *server configs* within a given period, which presents an indication for user tracking via the *server config*.

If log operators suspect a website to conduct tracking via the *server config*, they should recommend users to not reuse the *server config* for new connections and to conduct a full QUIC handshake instead. If the server submitted the respective *server config* to the log, then the log operator should sign the request with a dedicated private key to mark that *server config* as one that is suspicious to be used for user tracking.

This proposal mitigates tracking via *server configs* on a large scale, if the log operators allow only a single *server config* to be used within a specific period and mark all other *server configs* in the same period as suspicious.

5.2.2 Quick measures to protect against tracking via QUIC's token

In this section, we present steps that can be applied immediately by browser vendors and the QUIC working group to mitigate the privacy impact of the QUIC protocol.

As a first mitigation measure, **browser vendors should implement an expiry time for tokens**. The choice of an appropriate expiry time presents a trade-off between performance and privacy. A study of this trade-off based on real-world DNS traffic recommends limiting the expiry time to ten minutes [34]. Note, that on average, 27.7% of revisits to websites occur within this period. However, an expiry time of 60 minutes, allows in average 48.3% of website revisits to use 0-RTT handshakes, but a web tracker can observe a greater share of users' online activities in return. Furthermore, the results of the study suggest [34] that an expiry time longer than 24 hours does not significantly contribute to a higher share of 0-RTT handshakes. Thus, setting the expiry time to 48 hours instead of 24 hours increases the share of abbreviated handshakes only by 5.7 percentage points.

Furthermore, **a *server config update* message should not prolongate the feasible tracking period via tokens**. Instead, the expiry of the first token received from a web server should lead to a full handshake within the subsequent connection establishment, regardless of later expiry dates of tokens that were part of SHLO messages in the meantime.

The STTL hints on how long the server intends to support connection establishments for a given *server config*. Depending on the server implementation, it becomes more likely that a server rejects 0-RTT connection attempts after a *server config* expired. Anticipating this server behavior, it is beneficial from a privacy perspective to **conduct a full handshake after the respective cached *server config* has expired**.

A QUIC server uses the token to validate a client's IP address. Therefore, if a client changed the IP address after receiving a token, the server is more likely to respond with a REJ message during the next 0-RTT connection attempt. From a privacy perspective, the change of an IP address provides the opportunity to disassociate past activities from future ones, therefore breaking linkability at network-level. Thus, it seems reasonable to **establish new connections with a full handshake whenever a client's IP address changed**, to also disassociate activities in QUIC. Note that QUIC allows to explicitly migrate existing connections across different IP addresses. In this case, the server is already in a position to link both IP addresses to the same user based on the migration process. Thus, our proposal excludes this use case of connection migration.

To address the issue of third-party tracking via QUIC, browser vendors should restrict the access to cached *source-address tokens*. **Cached tokens of**

third-parties should only be used within the context of the same visited website.

5.2.3 Privacy-friendly connection establishment to protect against tracking via QUIC's token

We propose an alternative handshake design, that requires clients to only use tokens within their CHLO message if this message is a direct response to a server's REJ message. Figure 3 provides a sketch of these handshakes, where no longer required tokens are marked in bold and struck through.

This proposal applies strict source-address validation during the full and the rejected 0-RTT handshake, as shown in Figure 3 a) and c), respectively. However, the 0-RTT connection establishment does not conduct a validation of the source-address (see Figure 3 b). To mitigate excessive IP address spoofing, this design requires QUIC servers to monitor the number of unrequited connections. If the number of unrequited connections exceeds a specified threshold, the server falls back to an operation mode that rejects all 0-RTT connection handshakes (see Figure 3 c). Otherwise, the server allows 0-RTT handshakes without requiring a token.

By logical conclusion, this design is privacy-friendly because source-address tokens are not reused across different connections. Furthermore, it enables 0-RTT connection establishments whenever the number of unrequited connections are below the specified threshold. The definition of the threshold itself presents a trade-off between the protection against IP address spoofing and the performance of the connection establishment.

The original QUIC connection establishment requires the first visit of a website to be a 1-RTT full handshake because the client is required to retrieve a corresponding token, as shown in Figure 1. Assuming that a client is in possession of a corresponding *server config*, then our proposal allows establishing a 0-RTT handshake upon the first website visit. Therefore, our design saves a round-trip compared to the connection establishment of the original QUIC, if the server experiences a low number of unrequited connections and the client received its *server config* out of band. An out of band provision may take place via DNS. Thus, the client receives a valid *server config* before connecting to the website similar to the TLS extension *Encrypted Server Name Indication* [27], which distributes cryptographic keys via DNS.

6 Related work

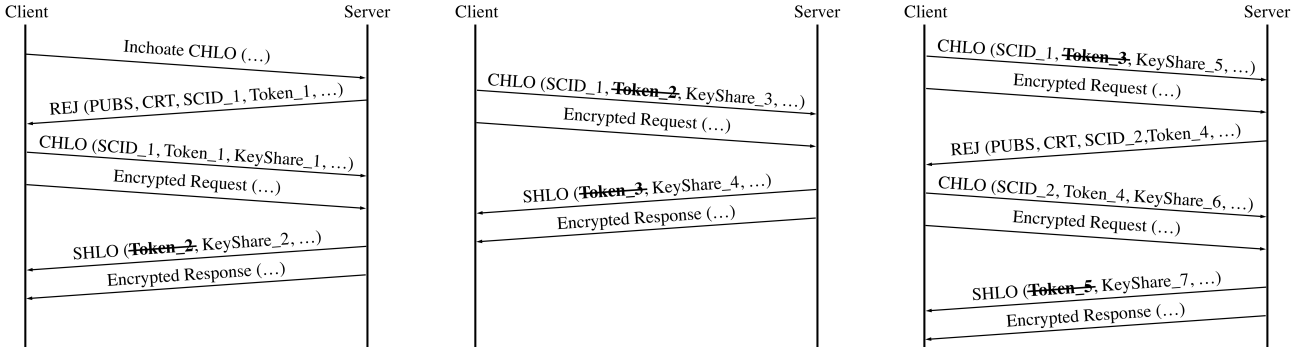
The security properties of QUIC have been discussed in prior work. A first cryptographic analysis of QUIC was done by Fischlin et al. [11]. Moreover, Jager et al. [13] analyzed the key exchange of QUIC and presented an attack on the confidentiality of the encrypted messages. Lychev et al. [21] demonstrated attacks against QUIC which lead to connection failures, server load, or server DOS.

In this work, we investigated QUIC version 39 which was negotiated by default between Google's servers and Chrome version 67 in June 2018. The IETF aims to redesign the key exchange of QUIC by adapting TLS 1.3 [12], which does currently not deploy a concept similar to QUIC's *server config*. However, the adaption of TLS 1.3 requires QUIC implementations to solve the privacy problem of TLS session resumption [34] which is not present within the current QUIC variant. Furthermore, the adoption of TLS 1.3 will not solve the privacy problems related to the *source-address token*.

We observe, that proposals such as OPTLS 1.3 [16] and the IETF draft on a semi-static Diffie-Hellman key establishment [26] envision to add a handshake design to TLS 1.3 which is similar to the discussed QUIC handshake and requires the long-term storage of a server's public key by the client. If adopted, this would lead to similar privacy considerations as discussed in the context of QUIC's *server config*.

Note, that the current IETF draft on QUIC [12] discusses the privacy consequences arising from the migration of QUIC connections across different IP addresses of a client. However, this tracking mechanism does not allow tracking across several user connections as our presented approaches do, because the ConnID which is used for the migration process is randomly chosen by the client at the beginning of each new connection.

To the best of our knowledge, we are the first to report on user tracking via QUIC across several connections and validated our findings based on current browser implementations. While there exist several web tracking approaches [6], the presented tracking mechanisms are exceptional for allowing unique user identification based on the first client message in an Internet-scale deployed transport protocol. Other tracking mechanisms via the HTTPS stack such as TLS session resumption or IP addresses require either a previous establishment of a TCP connection or do not provide unique user identification.



a) Full handshake b) 0-RTT handshake c) Rejected 0-RTT handshake

Fig. 3. Proposal for handshakes in the relaxed source-address validation scenario with no longer required tokens marked bold and struck through.

7 Conclusion

The presented tracking mechanisms via QUIC's *source-address token* and *server config* enable web servers to track their user. In the case of tracking via the *server config* even network-based attackers are able to identify clients across several consecutive connections. To the practical impact of tracking via QUIC contributes that widely used browsers such as Google Chrome support QUIC by default. Furthermore, our measurements indicate that these browsers do not employ protective measures against tracking via the presented approaches. Thus, also online third-party trackers can exploit these mechanisms and benefit from the potentially reduced delays compared to tracking via HTTP cookies.

The introduced countermeasures show that user tracking is not an inevitable by-product of realizing 0-RTT connections. We find, that the proposed privacy-friendly connection establishment for QUIC mitigates tracking via source-address token and furthermore directs QUIC to new performance gains. Our proposal allows a 0-RTT connection establishments upon the first visit of a website, if the *server config* is provided out of band, e.g. via DNS. Compared to the original QUIC protocol, this saves an extra round-trip during the first connection establishment.

In summary, we hope that our work leads to a greater awareness about the privacy risks in the QUIC transport protocol and spurs further research on user tracking via the HTTPS stack.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable feedback. This work is supported in part by the German Federal Ministry of Education and Research under the reference number 16KIS0381K.

References

[1] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689. ACM, 2014.

[2] A. Albasir, K. Naik, B. Plourde, and N. Goel. Experimental study of energy and bandwidth costs of web advertisements on smartphones. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 90–97. IEEE, 2014.

[3] Alexa Internet Inc. Alexa Top 1,000,000 Sites, 2018. URL http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[4] M. Bishop. Hypertext Transfer Protocol Version 3 (HTTP/3). Internet-Draft draft-ietf-quic-http-18, Internet Engineering Task Force, Jan. 2019. URL https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-18. Work in Progress.

[5] M. Boucadair, M. Ford, P. Roberts, A. Durand, and P. Levis. Issues with IP Address Sharing. RFC 6269, June 2011. URL https://rfc-editor.org/rfc/rfc6269.txt.

[6] T. Bujlow, V. Carela-Español, J. Sole-Pareta, and P. Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[7] G. Combs. Tshark- the Wireshark Network Analyser. *URL http://www.wireshark.org*, 2017.

[8] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security Symposium*, volume 8, pages 47–53, 2013.

[9] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1388–1401. ACM, 2016.

[10] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 289–299. International World Wide Web Conferences Steering Committee, 2015.

[11] M. Fischlin and F. Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1193–1204. ACM, 2014.

[12] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-Draft draft-ietf-quic-transport-12, Internet Engineering Task Force, May 2018. URL https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-12. Work in Progress.

[13] T. Jager, J. Schwenk, and J. Somorovsky. On the security of tls 1.3 and quic against weaknesses in pkcs# 1 v1. 5 encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1185–1196. ACM, 2015.

[14] C. K. Karlof, U. Shankar, et al. A Usability Study of Doppelganger, A Tool for Better Browser Privacy. 2007.

[15] G. Kontaxis and M. Chew. Tracking Protection in Firefox For Privacy and Performance. *CoRR*, abs/1506.04104, 2015. URL http://arxiv.org/abs/1506.04104.

[16] H. Krawczyk and H. Wee. The OPTLS protocol and TLS 1.3. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 81–96. IEEE, 2016.

[17] A. Langley and C. Wan-Teh. QUIC Crypto, 2018. URL https://www.chromium.org/quic.

[18] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, et al. The QUIC transport protocol: Design and Internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 183–196. ACM, 2017.

[19] P. Laperdrix, W. Rudametkin, and B. Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 878–894. IEEE, 2016.

[20] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, June 2013. URL https://rfc-editor.org/rfc/rfc6962.txt.

[21] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru. How secure and quick is QUIC? Provable security and performance analyses. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 214–231. IEEE, 2015.

[22] Y. Mansour, S. Muthukrishnan, and N. Nisan. Doubleclick ad exchange auction. *CoRR*, abs/1204.0535, 2012. URL http://arxiv.org/abs/1204.0535.

[23] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE, 2012.

[24] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of W2SP*, pages 1–12, 2012.

[25] Refsnes Data. The Most Popular Browsers, 2018. URL www.w3schools.com/browsers/.

[26] E. Rescorla and N. Sullivan. Semi-Static Diffie-Hellman Key Establishment for TLS 1.3. Internet-Draft draft-rescorla-tls13-semistatic-dh-00, Internet Engineering Task Force, Mar. 2018. URL https://datatracker.ietf.org/doc/html/draft-rescorla-tls13-semistatic-dh-00. Work in Progress.

[27] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood. Encrypted Server Name Indication for TLS 1.3. Internet-Draft draft-ietf-tls-esni-02, Internet Engineering Task Force, Oct. 2018. URL https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-02. Work in Progress.

[28] E. Roman and M. Menke. NetLog: Chrome's network logging system, 2018. URL https://www.chromium.org/developers/design-documents/network-stack/netlog.

[29] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld. A First Look at QUIC in the Wild. In *International Conference on Passive and Active Network Measurement*, pages 255–268. Springer, 2018.

[30] S. Schelter and J. Kunegis. On the Ubiquity of Web Tracking: Insights from a Billion-Page Web Crawl. *arXiv preprint arXiv:1607.07403*, 2016.

[31] P. Srisuresh and K. Egevang. Traditional IP network address translator (Traditional NAT). Technical report, 2000.

[32] StatCounter. Desktop vs Mobile vs Tablet Market Share Worldwide, 2018. URL gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide.

[33] StatCounter. The Most Popular Browsers, 2019. URL http://gs.statcounter.com/browser-market-share.

[34] E. Sy, C. Burkert, H. Federrath, and M. Fischer. Tracking Users Across the Web via TLS Session Resumption. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, pages 289–299, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6569-7. 10.1145/3274694.3274708. URL http://doi.acm.org/10.1145/3274694.3274708.

[35] The Chromium Project. QUIC, a multiplexed stream transport over UDP, 2018. URL https://www.chromium.org/quic.

[36] The Chromium Project. QUIC Wire Layout Specification, 2018. URL https://www.chromium.org/quic.

[37] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, page 3. ACM, 2013.