Čekerevac Z., DrSc, Full Professor, Faculty of Business and Law of the "Union – Nikola Tesla" University, Belgrade, Serbia
Dvorak Z., PhD, Professor, Faculty of Security Engineering of the University of Žilina, Žilina, Slovakia
Čekerevac P., MA, Hilltop Strategic Services, Belgrade, Serbia

HTTP* PROTOCOLS - PRINCIPLES AND NUMBERS

Annotation: *Although the Internet has been operating on the same basic principles for a long time, and even though it has billions of users, it is very rare that the user is aware of the functioning of the network. Therefore, at the time of increasing exposure to attacks, an Internet user is necessarily assigned to third-party services, and the network's operating principles and protection it considers as a black box. The aim of this article is to point out some principles of data transfer and to consider some dilemmas regarding the speed of data transfer and their security. The article presents the results of testing of HTTP, HTTP/2 and HTTPS protocols in various combinations and tests. There are shown and analyzed the results of authors' own research together with the results of other authors. The article ends with the conclusions drawn by the authors, that are based on the previous analysis and analysis of the results obtained.*

Keywords: *communications, interface, HTTP, Hypertext Transfer Protocol, HTTP/2, HTTPS, security.*

Introduction: The Hypertext Transfer Protocol (HTTP), as a stateless request/response application-level protocol for distributed, collaborative, hypermedia information systems, in its different versions is being successfully used since 1990. At the very beginning, HTTP/0.9, it was a protocol for raw data transfer, but starting with the version HTTP/1.0, it got many improvements, including capabilities to support transport of MIME-like messages [1]. Further improvements HTTP got in its version 1.1 with an open-ended set of methods and headers that indicate the purpose of the request [2, p. 359]. HTTP builds on the reference provided by the Uniform Resource Identifier (URI) [3], as a location (URL) [4] or name (URN) [5], for indicating the resource to which a method is to be applied [6, p. 293]. [7] This protocol is being in use for communication between users and their gateways, and other Internet systems that use SMTP, NNTP, FTP and other protocols, and the resources available from diverse applications. During its use, HTTP has seen many changes, from a protocol to exchange files in laboratory environment up to the modern Internet protocol which carries HD and 3D videos. More details about HTTP evolution can be found in [8]

Although HTTP/1.0 and HTTP/1.1 are widely used today, they have certain weaknesses relating to the number of outstanding requests at a time on a given TCP connection. Clients that need to make many requests use multiple connections to a server to achieve concurrency and to reduce latency. Also, HTTP headers are repetitive and extensive, and they can cause unnecessary network traffic, as well as the initial TCP congestion window quickly filling. This can lead to excessive latency when multiple requests are made on a new TCP connection. [9] HTTP/2 protocol is improved by optimizing of mapping of its semantics to an underlying connection. It allows prioritization of requests and interleaving of request and response messages on

the single connection. It uses much more efficient coding of HTTP header fields. By using binary message framing, HTTP/2 is more efficient in messages processing. Although at the application level HTTP/2 can be used as the stateless, HTTP/2 is stateful and it possesses stateful mechanisms. Having in mind all improvements, one can say that HTTP/2 protocol is friendlier to the network.

Hyper Text Transfer Protocol Secure[1] (HTTPS) is the secure version of HTTP and allows confidential online transactions on the safe way by using encryption, data integrity, and authentication via Transport Layer Security (TLS) protocol. HTTPS implementation demands security certificates issued by a certificate authority. And just on these certificates, the entire system is based. To make the system reliable, it is necessary that the user may believe that their browser with properly pre-installed certificates obtained from certification authorities correctly implements HTTPS protocol and that the certification authority guarantees only for legitimate Web sites issuing them valid certificates that accurately identify the website. Finally, it is very important that the encryption protocol (SSL or TLS) can prevent hacks.

HTTPs can be upgraded to secure operation by sending appropriate requests. Upgrading may be optional or mandatory. In the first case, a secure connection will be established whenever possible. The server completes current HTTP/1.1 request after switching to TLS. When an unsecured response must not be accepted, the client must send an OPTIONS request [10]. Within HTTP/2, secure connection establishes by using appropriate protocol identifier "*h2*"[2].

With the spread of the Internet and the increase in the number and value of transactions, the need for their protection increases. At the same time, it is necessary to ensure greater data transfer speeds. Further in this work, by comparing and testing, there will be discussed operational modes and performances of HTTP* protocols. Special attention will be paid to security aspects and ways of switching to HTTPS.

Overall Operation. As it was mentioned in Chapter 1, HTTP protocol is based on requests and responses. In the simplest case, data transfer can be accomplished using a single connection as shown in figure 1.

There are also HTTP communications that include one or more intermediaries

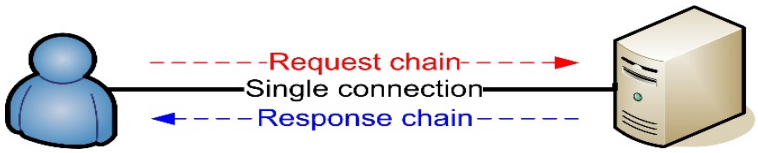


Fig. 1 The simplest case of HTTP communication

such as proxy, gateway and/or tunnel. Such transactions are more complex because they include receiving, rewriting, and forwarding requests, identifying the server, translating requests and tunneling. They are in some cases even more complicated because some options of HTTP communication may be applied only between two neighbors, other between the end points of the chain, or to all connections. Such an example is shown in figure 2.

[1] Also known as HTTP over TLS

[2] The string "h2" identifies the protocol where HTTP/2 uses Transport Layer Security (TLS) [21]

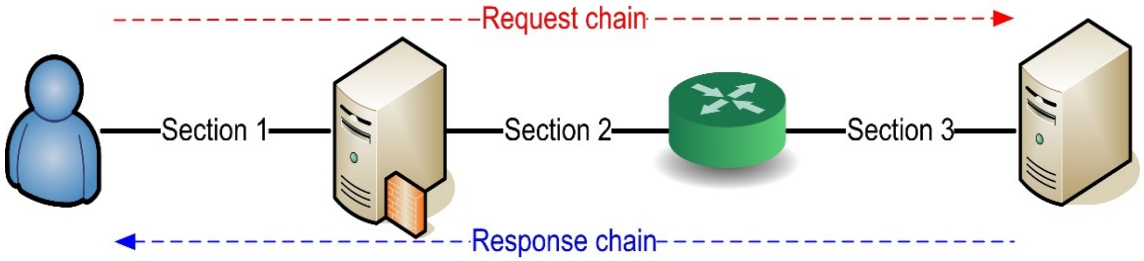


Fig. 2 An example of HTTP communication with a proxy and a gateway

In Figure 2 a communication seems linear, but in practice, this communication can be interrupted by other devices that also, and at the same time, want to communicate with some of the devices included in the chain, for example, proxy or gateway, router, etc. Each participant in the chain can have its own cache and can use it if the device doesn't act as a tunnel. In cases of using cache, some requests don't need to reach the end of the request chain, the response can be faster, and the whole network will be less loaded. In figure 2, if the response can be obtained from the proxy server, chain's sections 2 and 3, they won't be used. The response, in this case, is faster, but it does not mean that the user will get exactly what he expected if the cash is not up to date.

HTTP/1 communication usually, but not exclusively, uses TCP/IP over default TCP 80 port. This does not preclude HTTP from being implemented on top of any other protocol on the network, or to use another port. HTTP only presumes a reliable transport and beside TCP/IP, any protocol that provides such guarantees can be used. The mapping of the HTTP/1.1 request and response structures onto the transport data units of the protocol in question is outside the scope of HTTP specification [1]. In HTTP/1.0 for each request/response new connection establishes, but in HTTP/1.1 one connection can be used for more exchanges. Detailed specifications about HTTP/1.1 protocol can be found in [1].

HTTP/2 as an improvement of HTTP/1 protocol keeps supporting all its core features, but it allows many additional features which give significantly favorable characteristics to users. It is a frame based protocol. All frames begin with a fixed 9-octet header. The header is followed by a payload of variable length. Although variable, the default payload length is limited to 2^{14}, The receiver can allow larger lengths by setting a larger value, up to 2^{24}-1 octets. Each frame type has its specific task. Frames HEADERS and DATA are equivalent to the previous HTTP requests and responses. All other frames are used to support different HTTP/2 features. HTTP/2 allows multiplexing of requests through streams which are mostly independent of each other. Between new opportunities, HTTP/2 offers flow control and prioritization. They ensure that only useful data is transmitted to the receiver and that limited resources with the highest priority are forwarded to the most important streams. HTTP/2 communication by default, but not exclusively, uses port 80 for "HTTP" transactions and port 443 for HTTPS.

Precisely speaking, HTTPS is not a single protocol. HTTPS protocol identifier indicates that the HTTP protocol instead of via TCP uses over TLS. And, this is the

main concept of HTTPS. To establish HTTPS connection, it is necessary that HTTP client initiates a connection to the server on the appropriate port (as mentioned, usually, port 443). The next step is that the client sends the TLS ClientHello. After TLS handshaking, the client can initiate the first request, and send data as TLS "application data". When the data transmission is ended, and valid closure alerts are exchanged, the connection can be closed. In some cases, TLS implementation may send the closure alert as "incomplete close", and terminate the connection without waiting for peer's closure alert. Such situations can appear when the implementation wishes to reuse this session, or when the application explicitly knows that it has received all data of the message. According to RFC2246 [11] "any implementation which receives a connection close without first receiving a valid closure alert (a 'premature close') must not reuse that session" [12]. From the client side, any premature closes must be threatened as an error and received data as truncated. Such error can be in some cases tolerated, especially when it has received in the quantity specified in the Content-Length header. The client must send its closure alert before connection closing, but it can close the connection and require servers to recover transmission at any time. Servers should be prepared for different session endings and willing to resume such sessions. During the identification, it is assumed that the server hostname is known to the client, and that client must check whether server's Certificate message is sent from the expected server. This way it is possible to elude MITM attacks. Matching is performing by using the matching rules specified by RFC2459 [13] regardless of whether the URI is specified as a server's name or IP address of the server. If the hostname and the identity in the certificate do not match each other client must notify the user about that, or terminate the connection. The client must provide settings which can give an opportunity to continue with the connection. This method can't protect the transaction against attack when the source from which the URI comes is compromised. This weakness can be exploited for an MITM attack, as it was explained in [14]. From the server's side, it is not possible to check the client's identity, but if the server can identify the client, it should check its identity.

Performances. For the end-user, it is very important to have the web page in front of itself as soon as possible, not to perceive a latency, and to work interactively. An ideal protocol interaction can be if the client sends a request "GET/page" to a server, the server responds with sending enough data to the client, and the client gets the data to work without new requests on this point. HTTP/1 protocols are very good protocols, but their performances aren't the best they could offer. A typical transaction consists of several steps and it is rather "chatty". Clients send several requests with the header transmission to get a single web page. The things are even worse if the web page is not made as HTML only.

The HTTP/1.0 and 1.1 are discussed in many articles like [15], [16], [17], etc. Bearing in mind the imperfections of HTTP/1 protocols, we here will not consider each of them separately, and we will only present Nielsen's conclusions that "a pipelined HTTP/1.1 implementation outperformed HTTP/1.0, even when the HTTP/1.0 implementation used multiple connections in parallel, under all network environments tested. The savings were at least a factor of two, and sometimes as

much as then a factor of ten, in terms of packets transmitted. Elapsed time improvement is less dramatic and strongly depends on the network connection. [16]"
In this chapter, here will be compared some vital performances of data transmission using HTTP/* protocols. Such tests have been carried out by many authors. Also, there are a variety of the HTTP performance testing tools, such as Apache Bench (ab), Apache JMeter, Gatling, HP LoadRunner, WebLOAD, NeoLoad, OpenSTA etc. Experience shows that the performance impact of HTTPS is not a barrier to its adoption for websites. Our tests were conducted on the website http://www.httpvshttps.com/ that tests HTTP and HTTPS connections. During the tests, 360 unique and non-cashed images in a total of 0.62 MB were loaded. "Plaintext HTTP/1.1 is compared against encrypted HTTP/2 HTTPS on a non-caching, nginx server with a direct, non-proxied connection" [18]. In testing, it was observed a lot of dispersion of the results which can refer to the irregularity of the experiment itself, and interference of external influences.

Table 1.

HTTP vs HTTPS test results

Protocol	Time min	Time max	Time mean	Squares of deviations	Standard deviations	Variance
HTTP	5.96	12.56	10.52	34.04	1.945	3.782
HTTPS	3.47	9.52	6.50	33.57	1.936	3.730

Fig. 3 HTTP vs HTTPS test results

Results in Table 1 and Figure 3 claim that HTTPS loads significantly faster than HTTP. This is hard to believe because there are undoubtedly some overheads to HTTPS because HTTPS is normally applied on top of HTTP. However, it should be kept in mind that the different versions of the HTTP protocols were used and that HTTP/2 version caused favorable effects on using the HTTPS protocol. HTTP/2 includes header compression, and full headers should not be sent each time. This way for so many requests for small files like it was in this testing, a certain amount of time can be saved. In practical use, it is rather rare that a website loads so many images from the same place. From different locations access to the same website takes a

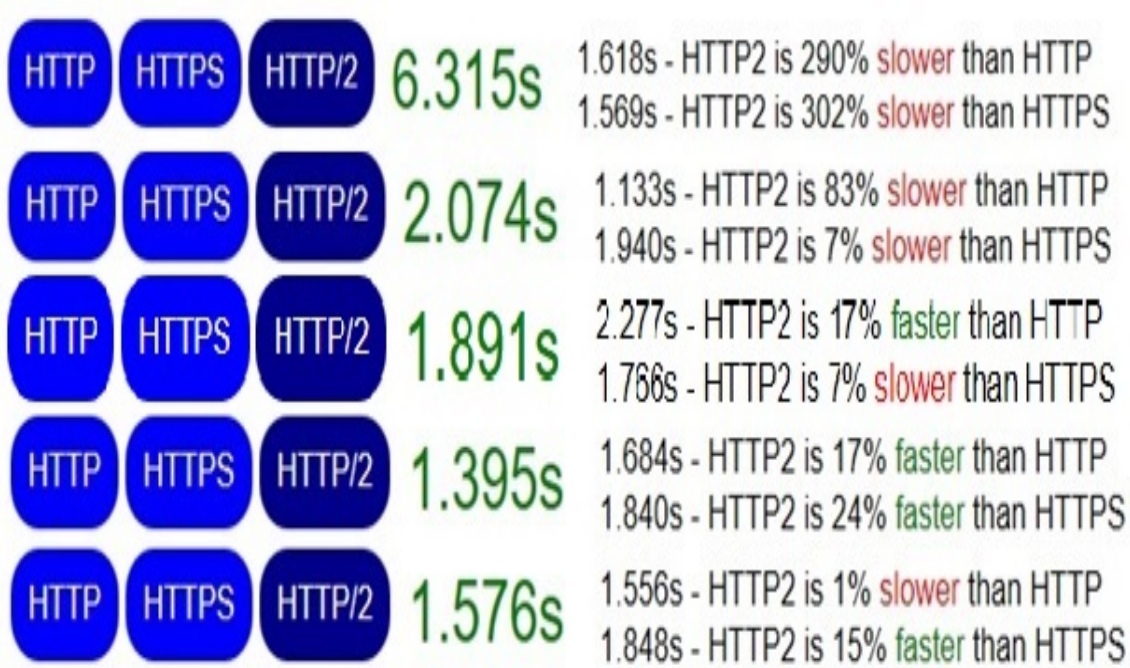
different time. An increasing number of advertisements also increases the boot time a web page. Therefore, very often, the user is not aware of the benefits that HTTP/2 brings.

Similar conclusions got Barry Pollard in his research published in [19]. He noticed that earned results are "an extreme example of a website that HTTP/2 is very good at - loading 360 pretty near identical images. While websites are continually growing upwards pretty much, no website loads 360 near identical images from the same domain - so that the only performance problem is latency (which HTTP/2 makes massive improvements on). Most websites will load a number of resources from multiple domains and some of those resources (particularly CSS and JavaScript) will take the time to process, whereas small images have negligible processing time." To give the more trustworthy test, Pollard wrote a similar test to compare HTTPv1.1 with HTTPS (over HTTPv1.1), and then with HTTP/2 (which uses HTTPS). He considered two cases: 36 images and 360 images. In our tests, we found that the successive tests vary considerably and that we for 36 pictures in 10 consecutive requests with HTTP/2 earned results in the range of 1.268s to 2.229s.

In his 36 images tests, Pollard got results shown in figure 4.

Fig. 4 Pollard's 36 images HTTP versus HTTPS versus HTTP/2 performance test

In our tests, using the same web site[3], we earned quite different results. Their examples are shown in figure 5. We repeated the test 60 times. The results are very diverse, from being HTTP/2 downloads much slower than HTTP and HTTPS, up to the cases when they are faster than HTTP and HTTPS. It is obvious that there exists also the impact of undefined external factors. These results suggest that the tests of



[3] https:

Fig. 5 Authors' 36 images HTTP versus HTTPS versus HTTP/2 performance test

this type are quite pointless. Valid testing can be done only in strictly defined laboratory conditions, but it seems as if such results are not significant for individual Internet users.

Significantly less dissipation of results we obtained in the same test[4], but in the case of transfer of 360 images. We repeated also this test 60 times. In 85% HTTPS (over HTTP/1) was slower than HTTP for 4.3% in average. In 98.3% HTTPS over HTTP/2 was faster than HTTP/1 and HTTPS over HTTP/1.

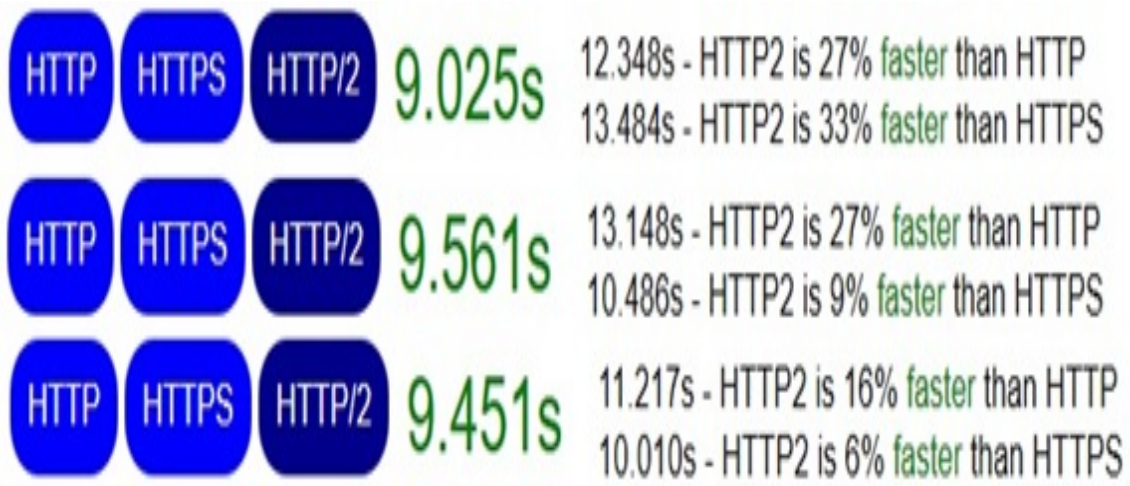


Fig. 6 Authors' 360 images HTTP versus HTTPS versus HTTP/2 performance test

Although there was a significant dissipation of the results, it looks like HTTPS over HTTP/1 cause no noticeable performance impact for simple websites, but, also, HTTPS over HTTP/2 alone don't bring significant speed advantages as expected. It has been stated by most experts for some time now. This, of course, can vary in different conditions, but, for most of the users, it is hard to notice the impact of HTTPS. Other influences like bad network connections, a poor mobile signal, distance from the server where the website is hosted, etc., are more important for a sense of speed for the user. In the most cases, the user doesn't know the cause of poor performances anyway.

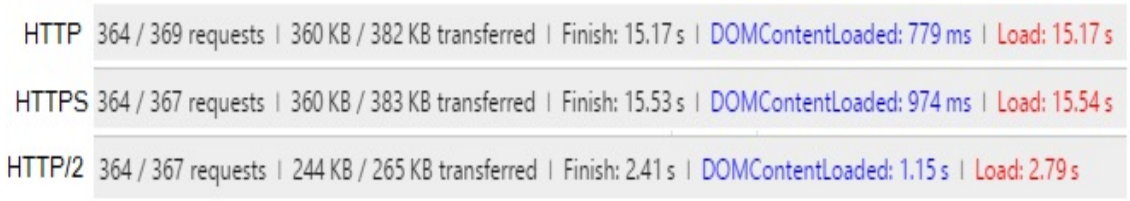


Fig. 7 Download characteristics of HTTP, HTTPS and HTTP/2
Source: [19]

In his research, Pollard found a significant download improvement using HTTP/2 because of the header compression. "In the 360 pages download with the identical total page size of 382kB in both HTTP and HTTPS drops to an impressive 265kb for HTTP/2 - a 30% improvement" [19], as it is shown in figure 7. However,

[4] https://www.tunetheweb.com/performance-test-360/

neither Pollard does not expect that a better result can be achieved in real terms, when larger files are placed on the website and when the impact of the header compression is significantly less.

Although small images were used in testing, and although most of the resources that are displayed on the websites are likely to be significantly larger, so the improvement due to the packing of the header will be less, however, the test results obtained are indicative, since they are obtained in real terms of exploitation.

According to Person [20], when Google applied HTTPS to secure emails between its users and Google, neither additional machines nor special hardware were deployed. On their "production frontend machines, SSL/TLS accounts for less than 1% of the CPU load, less than 10KB of memory per connection and less than 2% of network overhead" [20].

Conclusions. HTTP protocols are excellent technologies that enabled and still enable users to use the Internet, and computer networks in general, in a comfortable way. Based on the experience, and contemporary trends in computer communications, it can be concluded that most of the communications will soon be switched to HTTPS and that the HTTP sites will be suppressed. Switching to HTTPS for site owners may be tedious and costly, but it seems unavoidable, and it is better to do it as soon as possible. Each procrastination causes the problem to move at another time when the job of moving over will be even more complex.

HTTP/2 as a new technology is still not widely established, but, although the HTTP/2 specification was officially published in the middle of 2015, two years after, almost all web browsers support it. Also, many of web servers support HTTP/2, and it is to expect that this protocol will be the future of the web. Our tests showed how much faster it can be. The application of HTTP/2, with its multicast ability, compression of the packet headers, a more effective the SSL/TLS handshake, etc., it reduces the network load, which will be of great use in the application of future, increasingly demanding applications.

Finally, the application of HTTPS over HTTP/2 can be the win-win combination, it can give speed and security at the same time. Normally, work on the improving of the communication speeds and on the security, continues. Other protocols, including SSL/TLS protocols, they will be also improved, but not only they.

References

1. R. Fielding et al, "Hypertext Transfer Protocol -- HTTP/1.1," 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616.
2. L. Masinter, "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)," RFC 2324, 1998.
3. T. Berners-Lee, "Universal Resource Identifiers in WWW," RFC 1630, 1994.
4. T. Berners-Lee, L. Masinter and M. McCahill, "Uniform Resource Locators (URL)," RFC 1738, 1994.
5. K. Sollins and L. Masinter, "Functional Requirements for Uniform Resource Names," RFC 1737, 1994.

6. M. Hofmann and L. R. Beaumont, Content Networking: Architecture, Protocols, and Practice, Morgan Kaufmann, 2005.
7. N. Kew, The Apache modules book: application development with Apache, Upper Saddle River, NJ: Prentice Hall, 2007.
8. xcoderreal, "Evolution of HTTP," 9 Mar 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP.
9. M. Belshe, BitGo, R. Peon, I. Google, M. Thomson and Mozilla, "Hypertext Transfer Protocol Version 2 (HTTP/2)," Internet Engineering Task Force (IETF), 2015.
10. R. Khare, U. Irvine and S. Lawrence, "RFC:2817 Upgrading to TLS Within HTTP/1.1," ietf, 2000.
11. T. Dierks and C. Allen, "RFC2246: The TLS Protocol Version 1.0," Jan 1999. [Online]. Available: https://tools.ietf.org/html/rfc2246.
12. E. Rescorla, "RFC2818: HTTP over TLS," May 2000. [Online]. Available: https://tools.ietf.org/html/rfc2818.
13. R. Housley, W. Ford, W. Polk and D. Solo, "RFC 2459: Internet X.509 Public Key Infrastructure - Certificate and CRL Profile," Jan 1999. [Online]. Available: https://tools.ietf.org/html/rfc2459.
14. Z. Cekerevac, Z. Dvorak, L. Prigoda and P. Cekerevac, "Techno-economic aspect of the Man-In-The-Middle attacks," Komunikacie - Communications, vol. 19, no. 2, pp. 166-172, 2017.
15. S. E. Spero, "Analysis of HTTP Performance problems," July 1994. [Online]. Available: https://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html.
16. H. F. Nielsen, "HTTP Performance Overview," 06 Jan 2003. [Online]. Available: https://www.w3.org/Protocols/HTTP/Performance/Overview.html.
17. M. Nottingham, "Ideal HTTP Performance," 22 Apr 2016. [Online]. Available: https://www.mnot.net/blog/2016/04/22/ideal-http.
18. C. Anthum, "HTTP vs HTTPS Test," 2017. [Online]. Available: http://www.httpvshttps.com/.
19. B. Pollard, "Opinion - HTTP versus HTTPS versus HTTP/2," 22 Jul 2016. [Online]. Available: https://www.tunetheweb.com/blog/http-versus-https-versus-http2/.
20. K. E. Person, Interviewee, Overclocking SSL. [Interview]. 25 Jun 2010.
21. T. Dierks and E. Rescorla, "RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2," RFC Editor, 2008.