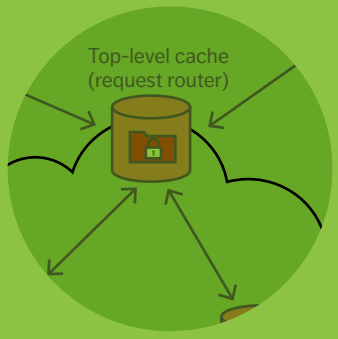
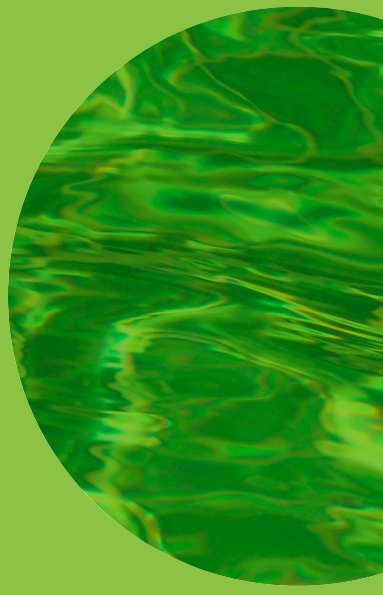
CHARTING THE FUTURE OF INNOVATION | #7·2016

ERICSSON TECHNOLOGY Review





BLIND CACHE
IN AN ALL-ENCRYPTED WEB

ERICSSON

Blind cache

A SOLUTION TO CONTENT DELIVERY CHALLENGES IN AN ALL-ENCRYPED WEB

GÖRAN A.P. ERIKSSON, JOHN MATTSSON, NILO MITRA, ZAHEDUZZAMAN SARKER

As the internet has shifted from an informal network for information sharing among universities and scientific institutes to a fundament of modern commerce, the traffic it carries has also shifted from short message exchanges to massive files encrypted for protection. At one time, the use of the secure communication protocol HTTPS tended to be limited to applications such as internet banking and online shopping. By the end of 2016, however, 70 percent of web traffic will use HTTPS [1]. Encryption is likely to become mandatory everywhere as the rise in pervasive surveillance has changed public perception on privacy and internet security, which has in turn prompted internet content providers to adopt HTTPS to protect consumers' content consumption data. The shift to pervasive encryption brings many benefits, but it also presents significant challenges for network service providers (such as mobile network operators and internet service providers) – particularly when it comes to caching content in their networks.

CONTENT DELIVERY networks (CDNs) improve delivery efficiency by replicating popular content like video streams on a cache serving users that are networked geographically nearby. The widespread adoption of CDNs has reduced latency and the amount of traffic carried by backbone networks. Existing CDN technology requires content providers to delegate their valuable content and expose traffic to the CDN provider, compromising end user privacy and security while revealing valuable business information.

> THE USE OF END-TO-END ENCRYPTION TAKES AWAY THE ABILITY FOR THE NETWORK SERVICE PROVIDER TO USE TRANSPARENT INLINE CACHES TO SERVE PREVIOUSLY REQUESTED CONTENT

By adopting HTTPS, the content delivery process becomes more robust, and both user security and privacy improve. However, the use of end-to-end encryption takes away the ability for the network service provider to use transparent inline caches to serve previously requested content, thus increasing backbone traffic as all requests for content have to be forwarded to and served by the content provider. Large content providers can sidestep this issue by placing content on their own edge servers, but this approach is costly and increases system complexity. Smaller content providers need another solution so that they can maintain low-latency content delivery and at the same time protect consumer privacy and ensure security. This need presents a business opportunity for network service providers to offer a better way to cache content that preserves the security and privacy of content providers and consumers.

To overcome the caching challenge inherent in HTTPS, Ericsson is collaborating with internet companies that have expertise in this area. Together we are exploring a solution we call blind cache, which is also referred to as out-of-band cache in industry discussions.

Encryption everywhere

There are very strong indications that the web is moving quickly toward HTTPS everywhere, or at least almost everywhere. This transition is driven by industry trends such as browser vendors providing HTTPS as the default option, warnings for non-HTTPS connections, and simpler and cheaper certificates and certificate management capabilities for smaller websites.

As *Figure 1* shows, the use of HTTPS has been rising since 2012. Further, it is greater in mobile networks than in fixed ones [1], owing to the growth of video mobile apps, which tend to use HTTPS by default. From a standardization perspective, security and privacy are the primary factors that

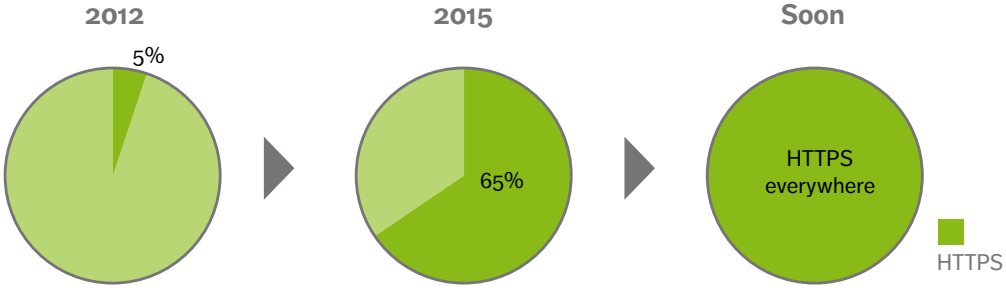


Figure 1
Evolution of HTTPS usage share in mobile networks

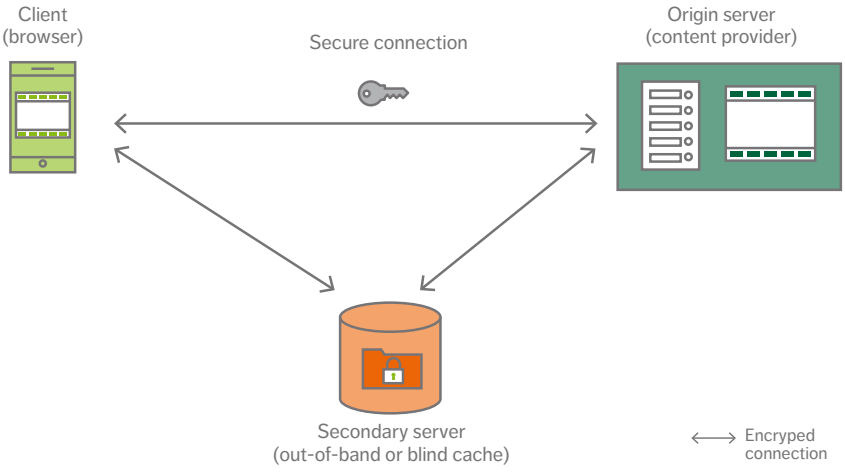


Figure 2
Blind cache architecture

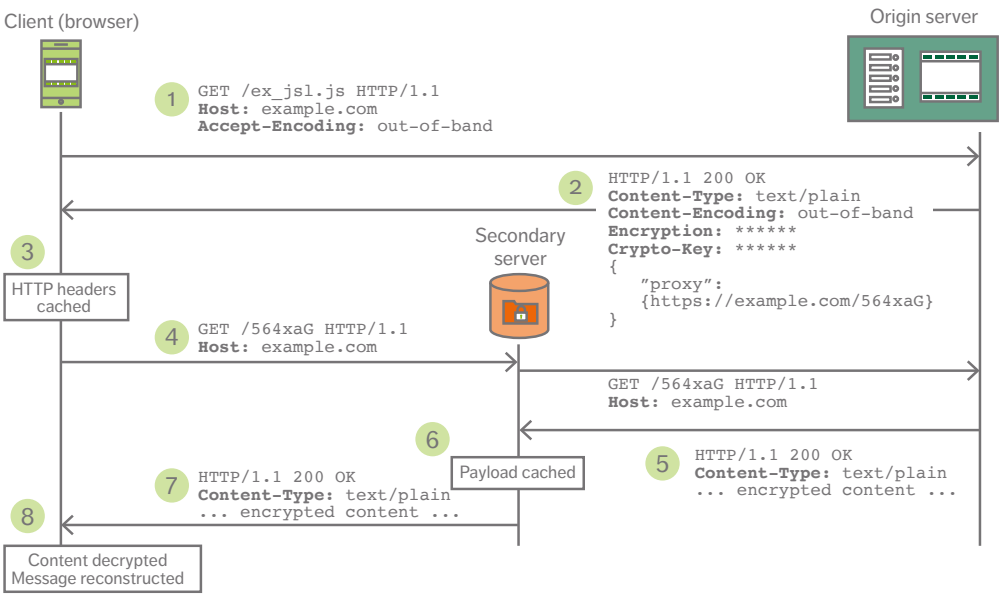


Figure 3
Blind cache call flow

need to be taken into consideration in developing the blind cache solution. Content providers, on the other hand, are primarily driven by the desire to secure end-to-end delivery, to protect ownership of valuable analytics data, and to protect against issues caused by network intermediaries such as ad injectors and application layer firewalls.

The web platform is transforming rapidly and the trend towards HTTPS is just one component of the ongoing evolution of the content delivery stack. On the client side, the industry is moving towards a new encrypted transport protocol evolving from QUIC [2], while content providers and CDNs are migrating to HTTP/2 [3] where the standard option is to mandate TLS or QUIC. The future content stack is HTTPS – HTTP or HTTP/2 over QUIC or TCP/TLS, so that a third party cannot read, alter, delete, insert, or replay data in any way.

Whether they like it or not, network service providers will have to adapt to HTTPS for all their traffic in the near future. As this requires preparation, it is time to start planning for their role in an all-encrypted web.

Solution concept

The philosophy of the blind cache solution is rooted in the concept of awareness; that all principal parties involved in a communication are explicitly aware of, and authorize the presence of, any intermediary participating in the data exchange. Adhering to this philosophy overcomes one of the main concerns about current, non-HTTPS, practice, in which the legitimate use of an intermediary such as an inline cache is made without the knowledge or consent of the user, and is indistinguishable from the actions of attackers. The target solution needs to ensure that service providers, such as a content provider, do not lose control of their content or other information flowing between client and server – a current drawback when content is served via a third party such as a CDN.

The blind cache solution [4] meets these requirements by creating a three-way relationship among the relevant actors, as shown in *Figure 2*.

It allows a content provider to deliver content faster by utilizing the support functions of external secondary servers for caching purposes. The solution places a requirement on the secondary server that any content cached on it remains encrypted and tamper-proof. It allows the content provider to decide if a secondary server should be used for a particular resource, such as an image, a JavaScript library or a set of video segments.

> “WHETHER THEY LIKE IT OR NOT, NETWORK SERVICE PROVIDERS WILL HAVE TO ADAPT TO HTTPS FOR ALL THEIR TRAFFIC IN THE NEAR FUTURE”

There are three scenarios available to a content provider for such delegated caches, with decreasing levels of trust.

Case 1 – edge origin

In this case, the secondary server hosting the blind cache belongs to the content provider. It is under the administrative and legal control of the content provider and is thus similar in character to the origin server.

Case 2 – CDN

In this case, the secondary server hosting the blind cache belongs to a third party – such as a CDN provider or a network service provider – with which the content provider has business and service level agreements.

Case 3 – proxy cache

In this case, the secondary server hosting the blind cache is hosted by any party known to the device, but does not require a business relationship with the content provider.

In all three cases, the content provider remains in charge of deciding what, if any, content to serve via blind caches. Since the caches are blind to the content they serve and unable to modify it,

> THE CONTENT PROVIDED TO A BLIND CACHE CAN BE ENCRYPTED, AND THE INTEGRITY OF THE CONTENT IS PROTECTED BY APPLYING SUITABLE TECHNIQUES

it is more likely that content providers will have the reassurance they need to use them. The blind cache solution also allows the content provider to delegate caching and serving of sensitive content to an untrusted server (or to a server on an untrusted site or cloud platform), or use a CDN provider that it might not have selected previously.

Case 1 is interesting given the possibilities for distributed cloud computing in mobile networks, which will lower the price point for a content provider operating an edge server remotely on a site and/or cloud platform under the control of a third party. An example of this is a mobile network operator that offers a cloud execution platform at a local central office site.

Case 2 enables a mobile network provider or a global CDN operator to provide deep edge-caching infrastructure.

Case 3 allows a client with a configured proxy – which is almost always the setup for enterprise users – to indicate its presence to the content provider. The content provider can then decide whether or not to allow the proxy to serve content on its behalf – restoring the efficiency gains of proxies in the service delivery chain, while preserving the security of the client-server communication.

It should also be noted that if the situation requires, the content provider could decide to have a virtual edge server serving only one user, a private cache. This could, for instance, be motivated when delivering a large file such as a software update to a particular enterprise customer, as an alternative to reserving a VPN connection to secure the timely delivery from a faraway origin server site.

Solution design

The message format used in the proposed technical solution is JSON, and the call flow is illustrated in *Figure 3*. The blind cache solution introduces the new parameter out-of-band [5] in the Accept-Encoding HTTP request-header field. Through this new parameter, a client indicates if it can (step 1 of Figure 3) handle HTTP responses where the payload is retrieved out-of-band (that is, from another server) separately from the main response. If the blind cache solution were to become well established, browsers would likely implement this feature and add the value by default in all requests.

If the content provider makes use of a blind cache, the HTTPS response from the origin server (2) will include the same out-of-band value in the Content-Encoding header field, informing the client that a cache will be used to deliver the payload of the response. The URL of the cache is included in the message body. Multiple URL may be provided, the first one linking to the primary cache with subsequent addresses linking to backups if the primary cache is inaccessible. The solution introduces two additional HTTP headers, Encryption and Crypto-Key, which contain the information for the client to decrypt the payload after retrieval, which the client stores (3).

In the next step, the client retrieves the payload from the URL provided (4). If the content does not exist in the cache, which may be the case when content is requested for the first time, the cache retrieves it from the origin server at the content provider (5), and caches it (6) for future requests. The content is sent back to the client (7) which can then decrypt the requested content (8) by applying the key acquired in step 3 to the encrypted payload obtained in step 7.

In deployment scenarios like edge origin and CDN (cases 1 and 2), content providers might want to pre-populate blind caches with content that is likely to be popular.

In addition to the call flow shown in Figure 3, the use of a client-selected proxy cache (case 3) can be achieved through the use of a proposed new HTTP header field with the working name BC [6].

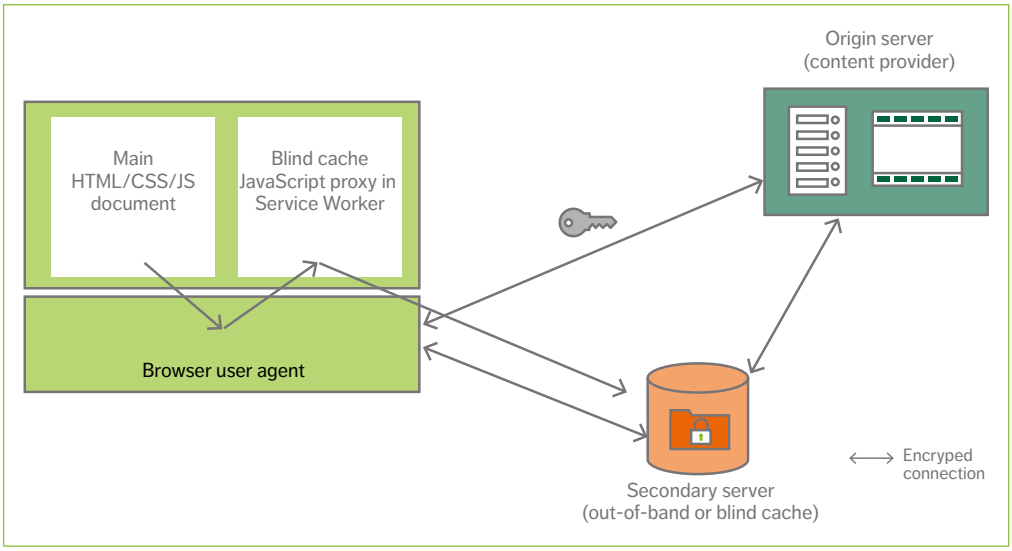


Figure 4
Overview of a browser-based client test environment

If present, BC indicates that the client is connected to a proxy cache that it is willing to use to retrieve content. If the content provider returns a response including the out-of-band value in the Content-Encoding header field, it accepts the use of that proxy cache (with which it has no relationship whatsoever) and is willing to delegate the handling of certain content to it.

The content provided to a blind cache can be encrypted, and the integrity of the content is protected by applying suitable techniques. One such technique ensures that resources at the origin cannot be inferred from delegated resources stored in a cache, and another prevents a cache from pretending to be a client and querying the origin, which could lead to discovery of the actual resources at the origin server [7].

Efficiency gains can be achieved through the use of a resource map that contains meta-information that the origin server can provide to the client. The resource map includes meta-information for all the delegated sub-resources – such as scripts, images, and video clips – that the client needs to create a complete representation of the set of cached resources. If the client and the origin server both support HTTP/2, the meta-information can be pushed to the client directly using the server push mechanism. Subsequent individual client requests for sub-resources can be redirected to the blind cache, which not only reduces latency for the actual request, but also decreases network traffic and processing costs at the origin server.

Tried and tested

Initial lab experiments conducted by Ericsson Research show that the use of a blind cache results in significant gains for serving a typical web page when the blind cache is pre-populated with the sub-resources or when the client is provided with the resource map – compared with retrieving the same content directly from the origin over a secure connection. As might be expected, the gain is greater when latency between the client and the origin server is longer.

To assess the HTTP extensions for blind cache, Ericsson Research developed a browser-based experimental testbed that is shown in *Figure 4*. The test environment uses the service worker

The tests require the client to fetch a test webpage directly from the origin server and the same content from the blind cache. The page load time is used as the key performance indicator (KPI) in the test results.

The blind cache can be **primed** (that is, it already has webpage resources), or **non-primed** (it does not have webpage resources and the resources are pushed from the origin server).

The client can also be in two states: **configured** – the Service Worker knows where to fetch contents, or **non-configured** – the resource map is not yet known.

The tests were run simulating different RTT values in the path between (1) the client and the origin server, and (2) the blind cache and the origin server.

Scenario A
RTT between the client and the server = 200 ms-300 ms
RTT between the client and blind cache = 40 ms
RTT between the blind cache and the origin server = 100 ms

Scenario B
RTT between the client and the server = 200 ms
RTT between the client and the blind cache = 40 ms
RTT between the blind cache and the origin server = 100 ms-200 ms

The results show that given high RTT between the client and the origin server, the proposed solution architecture will still be able to improve the user experience by a substantial margin (a page load time improvement of up to 30 percent). It also shows that the different delay between the cache and the origin server does affect the overall performance. The extra overhead required in terms of the number of extra bytes exchanged and extra request generated is also low compared with the gain in the responsiveness in page load.

The testbed helped identify different issues in the prototype and important features, which contributed to the evolution of the solution architecture and protocol design. The quest for more data and results continues; additional content types including video and delay scenarios will be added in the future.

Overhead due to use of BC

	% of extra bytes exchanged	Extra request generated
Primed, non-configured	3.03	3
Primed, configured	0.85	None
Primed, configured, all content via cache	1.29	None

Scenario A compared with end-to-end TLS

Cache primed?	Client configured?	All content via cache?	Page load time efficiency
RTT = 200 ms			
Yes	Yes	No	+27%
Yes	No	No	+11%
Yes	Yes	Yes	+38%
RTT = 300 ms			
Yes	Yes	No	+30%
Yes	No	No	+13%
Yes	Yes	Yes	+45%

Scenario B compared with end-to-end TLS

Cache primed?	Client configured?	All content via cache?	Page load time efficiency
RTT = 160 ms			
Yes	Yes	No	+41%
Yes	No	No	+14%
Yes	Yes	Yes	+39%
RTT = 200 ms			
Yes	Yes	No	+42%
Yes	No	No	+12%
Yes	Yes	Yes	+41%

mechanism [8] to implement a JavaScript-based proxy in the user agent that intercepts the HTTPS request from the main page and retrieves the resource map from the origin server as well as the origin payload from the secondary server.

When a response is received from the secondary server, the payload is decrypted and its integrity verified using the key provided by the origin server. Once the payload has been decrypted, subsequent requests for the webpage content are provided using the standard APIs in the Service Worker.

The content used in our tests included typical web resources, such as images, text, and DASH-segmented video.

Given the basic assumption that the latency between the client and the cache is lower than latency between the client and the origin server, tests were carried out to evaluate the proposed design for RTT – latency – in the various connections between the client, origin server, and secondary server (cache).

One potential disadvantage of a blind cache solution is the extra RTT required to retrieve out-of-band encoding meta-information from the origin followed by the content from the secondary server hosting the blind cache. To avoid this, the origin server responds to the client with out-of-band encoding information for a set of resources and not only the requested one. The tests were carried out with a view to improving the implementation of the protocol extensions, and to verify the assumption that a blind cache placed in close proximity to the client actually provides the desired benefits.

◀ PERFORMANCE TESTBED RESULTS

Key assumptions for the testbed:
a) Low bandwidth and high latency between the client and the origin server.
b) High bandwidth and low latency between the client and the blind cache.
c) The client and the blind cache may have the same access network characteristics towards the origin server.

> THE BLIND CACHE SOLUTION IS A SIGNIFICANT STEP TOWARD ENABLING CONTENT PROVIDERS TO LEVERAGE DEEPLY DISTRIBUTED EDGE CACHES WHILE MAINTAINING CONTROL OVER THEIR CONTENT AND ITS USE

Figure 5 is an example of a test scenario with different RTT measurements between client-origin, origin-cache, and cache-client in our testbed. In this particular scenario the page load time at the client improves by about 20 percent when fetching the web resources from the secondary server rather than fetching these directly from the origin server over HTTPS, thus confirming the benefit of using a resource map. As the client-origin RTT increases while other factors remain unchanged, the improvement in page load times rises.

A cloud of caches

The blind cache solution establishes the basic concept of placing and accessing content in a delegated cache in a secure manner. The next step is to explore possible ways to improve the efficiency of delivery, potentially using new deployment modes. *Figure 6* shows one area Ericsson Research is investigating: a set – or cloud – of caches.

In this hierarchy of caches, the top-level cache might belong to the same administrative domain as the origin server – the content provider, in other words. With the proper cache topology knowledge, the top-level cache can act like an HTTPS request router, redirecting requests for sub-resources to appropriate secondary caches based on a number of parameters, such as predictive traffic load, client location and topology related costs such as transport and CDN usage.

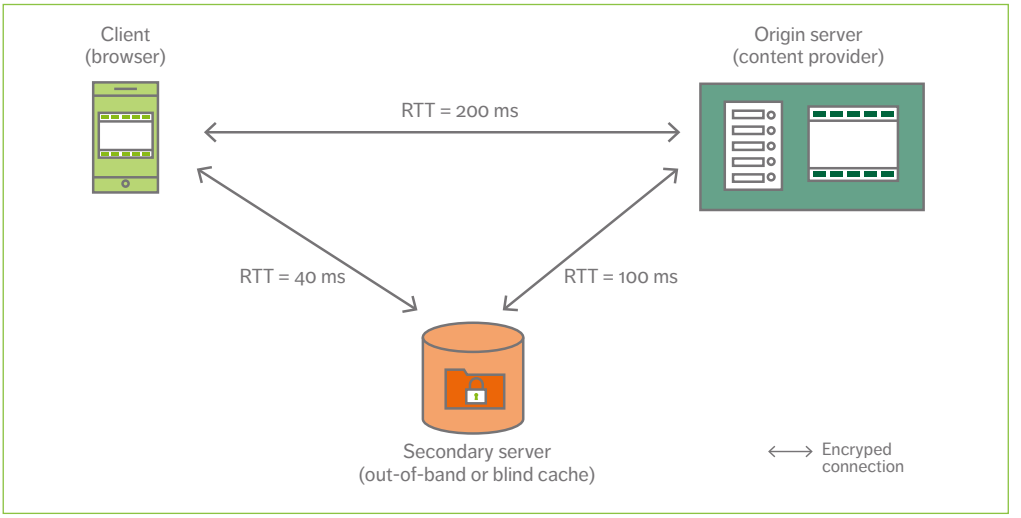


Figure 5
Example of a test with some sample latencies

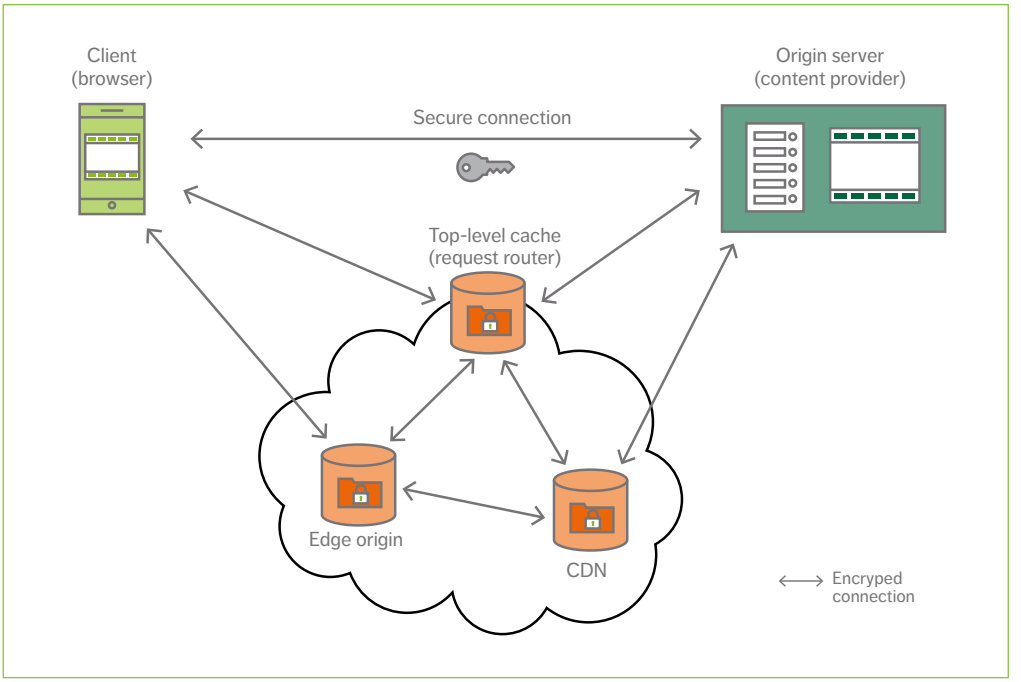


Figure 6
A set of caches

In Figure 6 the edge origin cache may belong to the administrative domain (the self-delegation case) of the content provider. The content provider could deploy such caches deep in a mobile operator's network – at a local central office site or a remote data center site, for example – leveraging shorter RTT and avoiding internet peering costs.

If the edge origin caches are provided by third parties for volume deployments in, for example, hotels, or to support Wi-Fi hotspots, the cost of provisioning certificates to secure the channel between the client and the caches can be reduced by provisioning the cache with inexpensive, self-certifying certificates such as those offered by the Let's Encrypt organization [9] and provisioned using the ACME protocol [10].

The benefits of using blind caches

The blind cache solution is a significant step toward enabling content providers to leverage deeply distributed edge caches while maintaining control over their content and its use. The solution provides network service providers with a business opportunity in an all-encrypted web, as it enables them to provide optimal placement of caches within their networks, as well as, for mobile operators, traffic management in the RAN that reduces latency and minimizes backhaul traffic. The solution will be particularly useful in 5G and LTE-U scenarios, where the need for caches will intensify – in residential gateways, hotspots, vehicles, and transportation systems – due to the dramatic increase in applications based on video traffic, which is projected to experience a 45 percent compound annual growth between now and 2021 [11]. ✱

Terms and abbreviations

ACME Automated Certificate Management Environment | **API** application programming interface | **CDN** content delivery network | **DASH** Dynamic Adaptive Streaming over HTTP | **HTTP/2** Hypertext Transfer Protocol Version 2 | **IETF** Internet Engineering Task Force| **JSON** JavaScript Object Notation | **LTE-U** LTE in unlicensed spectrum | **QUIC** Quick UDP Internet Connection | **RTT** round-trip time | **TLS** Transport Layer Security | **W3C** World Wide Web Consortium

References

1. **Sandvine, 2016, report, Global Internet Phenomena Spotlight – Encrypted Internet Traffic, available at:** *https://www.sandvine.com/trends/encryption.html*
2. **IETF draft, July 2016, QUIC- a UDP based Secure and Reliable Transport for HTTP/2, Work in Progress, available at:** *https://datatracker.ietf.org/doc/draft-hamilton-early-deployment-quic*
3. **IETF RFC 7540, 2015, Hypertext Transfer Protocol Version 2 (HTTP/2), available at:** *https://tools.ietf.org/html/rfc7540*
4. **IETF draft, June 2016, An Architecture for Secure Content Delegation using HTTP, Work in Progress, available at:** *https://tools.ietf.org/html/draft-thomson-http-scd*
5. **IETF draft, July 2016, 'Out-Of-Band' Content Coding for HTTP, Work in Progress, available at:** *https://tools.ietf.org/html/draft-reschke-http-oob-encoding*
6. **IETF draft, March 2016, Caching Secure HTTP Content using Blind Caches, Work in Progress, available at:** *https://tools.ietf.org/html/draft-thomson-http-bc*
7. **IETF draft, June 2016, Encrypted Content-Encoding for HTTP, Work in Progress, available at:** *https://tools.ietf.org/html/draft-ietf-httpbis-encryption-encoding*
8. **W3C, 2015, specification, Service Workers, available at:** *https://www.w3.org/TR/service-workers/*
9. **Let's Encrypt, available at:** *https://letsencrypt.org/*
10. **IETF draft, July 2016, Automatic Certificate Management Environment (ACME), Work in Progress, available at:** *https://tools.ietf.org/html/draft-ietf-acme-acme*
11. **Ericsson, 2016, Mobility Report, available at:** *https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf*

THE AUTHORS

Göran A.P. Eriksson

◆ joined Ericsson Research in 2000 and is an expert in communication services. He has worked on web technologies on the device and server side, ranging from SOA and composition engines to web protocols such as SIP, RTCWEB and HTTP to client side browser APIs – in particular WebRTC. He holds an M.Sc. in engineering physics from KTH Royal Institute of Technology, Sweden.

John Mattsson

◆ joined Ericsson Research in 2007 and is a senior researcher. In 3GPP, he has had a great influence on the work being done on IMS security and algorithm profiling. He coordinates Ericsson's

security work in the IETF, and is currently working on cryptography as well as transport and application layer security. He holds an M.Sc. in engineering physics from KTH Royal Institute of Technology, Sweden, and an M.Sc. in business administration and economics from Stockholm University.

Nilo Mitra

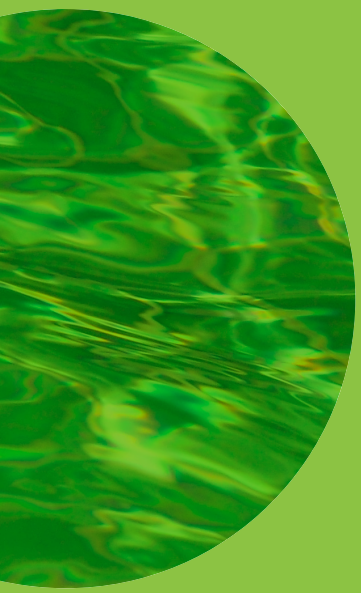
◆ joined Ericsson in 1998 after 15 years at AT&T Bell Laboratories. He is an expert standardization architect for media solutions at Business Unit Media, where he coordinates Ericsson's participation

in various media-related standards organizations. He has participated in many standardization fora including HbbTV, Open IPTV Forum, W3C, OASIS, WS-I, OMG, ITU-T, ISO, OMA and Streaming Video Alliance, having held leadership roles in several of these. Nilo holds a Ph.D. in theoretical physics from Columbia University, US.

Zaheduzzaman Sarker

◆ joined Ericsson Research in 2007 and is a senior researcher in the services, media and network feature domains. During this period, he has been involved in work from

prototype development to standardization, and from protocol design to radio network simulations (4G). His special areas of interest are real-time video communication, web technologies and performance measurements. He works with transport and application layer protocols in IETF, and is currently serving as Ericsson IETF coordinator. He holds an M.Sc. in computer science and engineering from Luleå University of Technology, Sweden.



ISSN 0014-0171
284 23-3285 | Uen

Ericsson
SE-164 83 Stockholm, Sweden
Phone: +46 10 719 0000