Beyond "Just TLS Everywhere": From Client-encrypted Messaging to Defending the Social Graph

Harry Halpin (W3C/MIT)
harry@w3.org

George Danezis (UCL)
g.danezis@ucl.ac.uk

Given the revelations of Snowden's leaks around the NSA and GCHQ, the strategic task that lies before the standards community is to find the minimal changes to the foundations of the Internet that prevent mass surveillance. No matter what the solution ultimately is it will have to be based on a mixture of encrypting content, fighting metadata collection, implementing data minimization, and preventing traffic analysis. So far, at the IETF on the HTTP level the focus has been to force "HTTPS Everywhere": to encrypt all Internet connections using TLS. However, is simply encrypting all internet connections really enough?

We argue that protecting all connections using TLS, as it is today, falls short of being a solution. First, the current implementation of TLS depends on trust in the CA system, which is flawed insofar as it is relatively easy to obtain a root CA certificate (particularly for a government) in order intercept HTTPS requests or impersonating a server. While certificate pinning combined with certificate transparency can mitigate the former and HSTS can mitigate the latter, nonetheless ultimately the NSA revelations have shown that a powerful enemy can simply request the unencrypted data from the server through legal compulsion, regardless of any encryption on the level of the network. We consider the primary goal of stopping pervasive surveillance making it technically as difficult as possible for the contents of a message to be read by *any party* other than the intended recipient, and so disagree with much of the response of Perens to the goals of stopping pervasive surveillance by distinguishing between government and non-governmental interception. Therefore we argue standards bodies must also provide solutions beyond merely client-server secure tunnels.

What is necessary is end-to-end, i.e. client-to-client, encryption applied to the actual content of messages. The private key material must remain unknown to any server, so that it cannot access the encrypted message under compulsion. Take the example of SMTP email: Surprisingly, an open source encrypted email client and server system based on client-side encryption that is simple for both users and system administrators to use has not yet been built. While a number of protocols have been developed by the IETF to provide the capabilities for encrypted email in general they are not widely deployed. The problem can be broken down into a number of distinct components: server-side infrastructure, usable client software, and the fundamental protocol itself.

We outline how a new open-source code project for encrypted email, the LEAP Encryption Access Project[1] is tackling these problems, and how a number of open issues remain, ranging from perfect forward secrecy of asynchronous messaging to maximizing the use of mix networks to avoid traffic analysis.

Current OpenPGP-compliant or other content encryption protocols are difficult to set-up for many end-users and server administrators. As a result only a few large servers such as Google or activist e-mail servers under considerable threat (such as `riseup.net`) implement the server-side infrastructure for the protocols such as proper use of DKIM, and while standards like S/MIME are more widely supported, they are ignored by webmail clients and even many non-Web clients. LEAP facilitates such infrastructure deployment by creating "puppet" (automation) scripts for many of the harder tasks involved in setting-up a privacy-enhanced and secure email provider. However, setting up certificate pinning and other best practices for email service providers is not enough. What is necessary is to have the client and server actively work together in order to encrypt the message, to prevent the situation where private key materials are stored only on the server and defended only by weak defenses such as passwords. This was the case for Lavabit[2] with unfortunate consequences.

As has been amply demonstrated by Greenwald's fail-

[1] `https://leap.se`

[2] `http://www.thoughtcrime.org/blog/lavabit-critique/`

ure to successfully set up encrypted email to communicate with Snowden, current encrypted email is simply too difficult for users. Much of the problem comes not only from poor user experience, but also from the fact that almost all encrypted email systems force all key management onto the user. Naturally, the user should not have to refer to themselves as an indecipherable bit-string nor engage in cryptic operations involving keys: We argue that any encrypted messaging system should aim for *key invisibility*, where the key itself, but also key management is invisible to the user. The main problem facing such a system is safely getting the correct keys onto users devices.

LEAP accomplishes this through the LEAP client that serves both as an OpenVPN client and as a tool for generating, validating, and discovering keys as well as synchronizing keys and related material (such as the status of messages being "read" across multiple devices). The LEAP client piggybacks on an OpenVPN since many users do install a VPN on their system to enable activities such as file-downloading or watching streaming videos. This OpenVPN LEAP client can then also generate keys, and with the help of the server even manage the keys by using server-enabled discovery and trusted validation of public keys for the recipient of email. Lastly, while the new email client natively supports encryption, that could replace Mozilla Thunderbird and Enigmail, LEAP allows users to continue using their existing non-Web e-mail client: it provides a local SMTP proxy that captures unencrypted email, encrypts it, and then sent it out using the LEAP protocols.

The LEAP system client and server are open-source and non-profit, and we hope will lead to a network of surveillance resistant e-mail providers emerging to help tackle pervasive surveillance. The project source-code on Github is available to all for security reviews and can be downloaded for beta-testing at Bitmask.net [3].

It should be obvious that encrypted network connections via TLS is not enough, we must encrypt the full message itself from end-to-end. However, in many cases even end-to-end encryption is not enough. Users deserve encrypted messaging *now*, even if we cannot solve all their problems by creating the "perfect" system. However, even content-encrypted messages would not prevent the mapping of the social graph, one of the most dangerous techniques of mass surveillance. A number of problems must then be solved: namely, how do we disguise the sender and the recipient in an encrypted email? How can the email be sent to groups while maintaining perfect forward secrecy? And lastly, how can we use constant-time and padding to prevent traffic analysis?

It has been clear for some time, and the recent revelations confirm this, that the backbone of signals intelligence is based on traffic analysis – who talks to whom. It is no surprise that signaling protocols have been the subject of interception, and that address book information from major providers has been intensively mined by intelligence agencies. However, unlike protecting content through encryption, protecting meta-data is much harder since it is relied upon to make network protocols more efficient. Communications meta-data is in particular required in two key settings: *name translation*, and *routing*.

Name translation is used all the time in network protocols to turn high-level names, such as domain names, into low level addresses such as IP addresses. Such lookups, even when protected through encryption have the potential to leak information to network observers about who is talking to whom (for example by observing the traffic patterns of a recursive DNS resolver). Name resolution does usually require asking a server for name translation making it inherently more vulnerable to compulsion. Similarly, querying key servers, or the mere presence of a user, also leaks information about who the intended recipient of a message may be despite any encryption. The LEAP project is currently working with security researchers to design protocols that can be used to securely perform these tasks, without leaking any or at least as much information about users' social graphs to servers.

Routing information is the ultimate source of signals intelligence since it provides a wealth of information, but seems hard to patch: messages need to be routed through a distributed system like the Internet or SMTP servers, by intermediate routers that are inherently untrusted. Solutions to this problem involve mix-network based routing for high-latency traffic, or incorporating ideas from Onion Routing into lower level internet protocols. How to do these efficiently is an open research problem, and while LEAP would like to implement a solution to this problem, it is not clear that any current design (including the state of the art anonymizer Tor) could scale to serve a very large fraction of the Internet.

In the long-term, we would like encrypted messaging to be usable on the Web. Only by creating a Javascript Web Cryptography API that can successfully prevent the private key material from being accessed by the server can we successfully allow encrypted messaging to be boot-strapped on top of existing Web clients rather than having to use native applications such as OpenVPN clients mixed with non-Web mail clients. Ultimately, anonymity loves company - for us to be successful in fighting the surveillance by the powerful secret state, our technical solutions must be majoritarian: they must serve all the people, and be cheap and usable enough to represent the default way of communicating.

[3] `http://www.bitmask.net`