Is Opportunistic Encryption the Answer? Practical Benefits and Disadvantages

John Mattsson
Ericsson Research

Abstract. Following the recent pervasive monitoring revelations, one of the discussed remedies has been opportunistic encryption. In this paper, we give an overview of various opportunistic and unauthenticated encryption techniques, discuss their benefits, limits, and disadvantages, and try to conclude how effective they are as a mean to thwart pervasive monitoring.

1. Introduction

Following the recent pervasive monitoring revelations, one of the discussed remedies has been opportunistic encryption. The term opportunistic encryption is, however not very well defined and has been used for very different systems, only having in common that something in the setup is opportunistic. The term opportunistic encryption is sometimes used as a synonym to unauthenticated encryption, meaning that the endpoint is opportunistic about the other endpoint being the desired endpoint and not an adversary. But opportunistic encryption can also mean that the endpoint is opportunistic about the other endpoint's support of encryption (falling back to unencrypted otherwise). In most cases, it refers to a system where encryption is added (sometimes or always) without any pre-deployment of credentials between the two endpoints (at least not for the protocol in question), and where the endpoints do not authenticate each other. The end result is that the endpoints can be unauthenticated, weakly authenticated, or strongly authenticated, in some cases without the endpoints even knowing which.

Whereas encryption traditionally has only been used where necessary and when the end user specifically asks for it (secure email, TLS, crypto phones) or when set up by a network administrator (IPsec between sites), one could say that the aim of opportunistic encryption is to encrypt whenever possible. Similar to [1], we define opportunistic encryption to mean systems where encryption (authenticated or unauthenticated) is automatically set up (if supported) without user requests or administrator involvement. We define unauthenticated encryption to mean systems where the endpoints are not strongly authenticated and where man-in-the-middle attacks may be possible.

2. Deployments of Opportunistic Encryption

The most common and successful usage of opportunistic encryption has been for server-to-server encryption, and the most widely deployed and used system is probably STARTTLS for SMTP [2], which opportunistically tries to upgrade a plaintext SMTP connection to an encrypted connection over TLS. Another example is the swan projects (FreeS/WAN, Openswan, and Libreswan [3]) that rely on DNS for dynamic discovery and negotiation of IPsec tunnels between gateways. If DNSSEC is used, the encryption is authenticated and secure also against active attacks. In these use cases, the end user is totally unaware of the encryption.

Opportunistic encryption for client-server or client-to-client communication is not widely deployed. One client-to-server system is HTTPS Everywhere [4] which enables HTTPS where supported.

3. Methods for Unauthenticated Encryption

Unprotected key transport. The simplest way to achieve unauthenticated encryption is to just transfer the session key or password in the hope that the communication channel was either confidential or that no adversary happened to eavesdrop. Examples are SDP security descriptions [5] and the extremely common use of unprotected emails for sending out new or forgotten passwords. In these cases, the hope is that the key transport is at least protected hop-by-hop and in the email case that the legitimate user changes the temporary password before any eavesdropper do.

Compared to unprotected communication, this approach forces an eavesdropper to decrypt the communication and to intercept different message types potentially taking different routes. Except for the case with temporary passwords, an adversary can be completely passive and cannot be easily detected. The eavesdropper may store the encrypted communication for later decryption. In the case of temporary passwords, an adversary needs to act quickly before the legitimate user changes the password.

Self-signed certificates. A more sophisticated type of unauthenticated encryption uses self-signed certificates (or raw public keys) in a key agreement protocol. One example is DTLS-SRTP [6] with a non-Diffie-Hellman ciphersuite.

Compared to key transport, an eavesdropper is forced to be active, but as the eavesdropper can typically negotiate the same session key with both endpoints the eavesdropper can be passive after the initial key management phase. The eavesdropper cannot easily be detected except by using some out-of-band channel (see Section 4) to compare the exchanged certificates.

Diffie-Hellman key exchange. The most secure type of unauthenticated encryption uses a Diffie-Hellman key exchange (with or without exchange of self-signed certificates). Examples are SSH, the Tor project [7] (that provides anonymity but not end-to-end confidentiality), and DTLS-SRTP [6] with a Diffie-Hellman ciphersuite.

Compared to non-Diffie-Hellman key agreement, the adversary cannot negotiate the same session key with both endpoints forcing the adversary to decrypt and re-encrypt all messages in real-time. It also provides perfect forward secrecy.

4. Weak Authentication Methods

Unauthenticated encryption is often strengthened by some form of weak authentication [8] that gives imperfect, but -in some cases- good enough, security. By weak authentication, we mean authentication without relying on trusted third parties and without physically meeting and exchanging credentials.

Out of band. Ensures that the same endpoint is reachable via more than one communication path. Examples are SDP security descriptions, DTLS-SRTP, the common use of email for sending out new or forgotten passwords, and the use of SMS text messages to deliver one-time passwords. An attacker needs to intercept both paths, and potentially also do active attacks.

Locality check. Ensures that the other endpoint is relatively close, e.g. by measuring round-trip latency. One example is HDCP v2.x. Limited round-trip latency also limits the amount of processing an adversary can do, which may increase security.

Key continuity. Ensures that the other endpoint is the same endpoint as in the initial connection, by checking that is has access to the same credential as before. Examples are SSH, DTLS-SRTP, and CDMA over-the-air provisioning. An attacker needs to be present during the first key management session.

Human interactive proof. Test used to determine whether the other endpoint is human or not. One example is short authentication strings used in crypto phones and ZRTP [9]. Short authentication strings may also provide authentication by voice recognition.

All these methods have weaknesses but e.g. key continuity may be excellent when the first communication is on a trusted local network. Short authentication strings between people knowing each other have traditionally been used in crypto phones, but agencies are now believed to have developed experimental voice analysis and synthesis systems [9].

As described, the so-called “weak” authentication may give high assurance and similarly the so-called “strong” authentication methods may give low assurance when used incorrectly. The classification into strong and weak authentication methods is therefore far from optimal and it may be better to talk about the level of assurance in the authentication.

5. Benefits against Pervasive Monitoring

While the use of unauthenticated encryption increases the cost of pervasive monitoring, we estimate that the cost increase is quite low. Just as adding encryption and authentication to a protocol usually only increases the CPU usage by a small percentage, the cost increases for an eavesdropper forced to decrypt, or decrypt and re-encrypt is low. An eavesdropper doing pervasive monitoring is already saving the information or at least scanning the messages for relevant information, adding some cryptographic processing which could be done in hardware is not going to be a major obstruction. An evaluation in [10] shows that the throughput of an SIP proxy with TLS is slightly below half that of a SIP proxy with unprotected TCP. We estimate that an agency doing pervasive monitoring would face a similar performance penalty, and that the use of dedicated hardware can shrink this difference even further.

On the other hand, it is likely that an agency doing pervasive monitoring, and especially active attacks, places a high value on not being detected. Unauthenticated encryption with Diffie-Hellman key exchange would enable detection of man-in-the-middle attacks. To reveal pervasive monitoring on a massive scale it would only be necessary to manually compare session keys out-of-band on a regular basis.

6. Disadvantages

Introducing opportunistic or unauthenticated encryption in protocols is likely to introduce yet another option, therefore increasing complexity. It's a fact that the secure modes of many protocols are often not implemented, implemented later, or not well tested. Assuming that developers will suddenly begin implementing two different secure modes is not realistic.

There is also a risk that developers, people deploying systems, and end-users (wrongfully) think that unauthenticated encryption is good enough, therefore reducing the implementation and use of strong authentication. In fact, very few people deploying or using the systems are likely to understand what kind of security is really offered, and under which circumstances they are protected against various attacks. One example of an extremely complicated trust model is DTLS-SRTP [6], which on top of all ciphersuites and key management options in DTLS and SRTP adds two forms of weak authentication in the form of certificate fingerprints and key continuity. Average end users can in most cases not be expected to do much more than to distinguish between secure or not secure (e.g. by looking for a padlock icon), and nor should they.

7. Conclusions

Opportunistic and unauthenticated encryption certainly has its uses and may with the right choices provide good enough security for a low cost. The use of opportunistic encryption in STARTTLS for SMTP and the use of key continuity as a weak authentication method in SSH are excellent examples of that.

However, we do not think that massive deployment of opportunistic and unauthenticated encryption is a general solution to the pervasive monitoring problem, and may even cause as much harm as good. The additional cost for processing and memory is likely quite low, or at least not high enough to substantially reduce the amount of pervasive monitoring. However a high risk of being detected might refrain agencies from pervasive monitoring. But even if it's possible to identify that there is an active man-in-the-middle, it might still be difficult to identify where it's located and who it is.

Our opinion is that any recommendation should not be stressed and that the Internet community should take its time to think and discuss before agreeing on any general BCP for opportunistic and unauthenticated encryption.

Our recommendation is that unauthenticated encryption is only used when some form of weak authentication is likely to be available. For opportunistic encryption we think that the impacts such as operational and network management aspects needs to be analyzed and understood for each protocol and that general recommendations are therefore hard to make.

We recommend the Internet community to clearly define the term "opportunistic encryption" or to use other terms. If encryption without authentication is meant, we

suggest that “unauthenticated encryption” is a better expression. If used, the number of options should be reduced and that the implementation and naming should not lead the end user to believe that the security is better than it really is.

References

[1] Citizendium, ”Opportunistic Encryption”, http://en.citizendium.org/wiki/Opportunistic_encryption

[2] IETF, RFC 3207, “SMTP Service Extension for Secure SMTP over Transport Layer Security”, 2002

[3] The Libreswan Project, https://www.libreswan.org/

[4] EFF, HTTPS Everywhere, https://www.eff.org/https-everywhere

[5] IETF, RFC 4568, “Session Description Protocol (SDP) Security Descriptions for Media Streams”, 2006

[6] IETF, RFC 5764, “Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)”, 2010

[7] Tor Project, https://www.torproject.org/

[8] Arkko, Nikander, “How to Authenticate Unknown Principals without Trusted Parties”, 2002, www.ietf.org/proceedings/57/slides/enroll-4/enroll-4.ppt

[9] Wikipedia, ”ZRTP”, http://en.wikipedia.org/wiki/ZRTP

[10] Shen et al. ”The Impact of TLS on SIP Server Performance”, 2010, http://academiccommons.columbia.edu/download/fedora_content/download/ac:127663/CONTENT/cucs-022-09.pdf