# It Is Time to Redesign Transport Layer Security

Francisco Corella, PhD
fcorella@pomcor.com

Karen Lewison, MD
kplewison@pomcor.com

Revised and expanded: January 12, 2014

**Abstract**

The current transport layer security protocols, TLS and DTLS, have serious security, privacy and usability shortcomings that cannot be practically fixed by incremental upgrades. We argue that the current awareness of the breakdown of privacy and security on the Internet provides an opportunity to redesign transport layer security from scratch. We list several ingredients that could by used in such a redesign, including the use of multi-root, hierarchical identity-based encryption for key exchange, and the dissociation of client authentication from the key exchange to allow for an open-ended variety of client credentials and for the implementation of login sessions without cookies. Successful deployment of the redesigned protocols would substantially increase security and privacy IP networks.

## Contents

# 1 Introduction

The recent revelations of massive bulk surveillance by several governments have been shocking. It is sad to hear that the US Government is purposefully weakening Internet security [1, 2].

But they also come with a silver lining. They have made everyone more aware of the breakdown of privacy on the Internet as not only governments but also private corporations spy and build dossiers on Internet users [3]. This awareness may provide the momentum needed for revamping the technological foundations of security and privacy of the global network. Indeed, the very same governments who conduct surveillance must protect themselves from spying by their own adversaries, and they are actually funding research on Internet anonymity [4].

Now is the time to tackle the privacy and security shortcomings of network technology. One aspect of network technology that needs revamping is transport layer security. Transport layer security protocols are interposed between the transport layer and the application layer of a network stack to provide protection for data transported between application endpoints over a potentially compromised network. They provide security features typically including data confidentiality, data authentication and entity authentication for one or both endpoints.

On the Internet, which, as we now know, is a massively compromised network, transport layer security is provided by TLS [5, 6, 7] for data transmitted over TCP connections, and DTLS [8, 9] for data transported by UDP datagrams. TLS is by far the most widely used network security protocol, as it protects access from web browsers to `https` URLs, and provides underlying security for many other Internet protocols besides HTTP. DTLS provides security for protocols such as SIP, which is used to control voice and video calls over IP networks.

TLS and DTLS have serious security, privacy and usability shortcomings, discussed below in Section 2, including: cryptographic vulnerabilities; lack of support for anonymous credentials and poor support for other cryptographic client credentials; and usability shortcomings that prevent ubiquitous deployment, viz. high latency over satellite links caused by handshake roundtrips, and excessive consumption of scarce bandwidth for transmission of certificate chains over satellite and radio links. Governments, specially their military branches, should be particularly concerned about latency and bandwidth consumption. The US military, for example, has its own private IP network, the *Global Information Grid* [10], which relies heavily on satellite and radio links.

It would be impractical to remedy these shortcomings by modifying the existing protocols, because of the scope of the remedies, and because of increasing resistance to upgrade. TLS and its predecessor SSL are old by computing technology standards, since they date back to the release of SSL 2.0 by Netscape in 1995 [11]. In their long history they have gone through five versions, four of which are in use today, and they have accumulated a large number of extensions. The proliferation of versions and extensions makes it difficult to configure implementations that provide both security and interoperability, and causes operators of TLS servers to resist upgrades. A survey of a sample of web sites that support TLS and/or SSL [12] shows that, as of November 2013, only 20.7% had deployed TLS 1.2, specified in August 2008, and only 30.6% provided server-side protection against the BEAST attack [13], which requires either TLS 1.1, specified in April 2006, or TLS 1.2.

It is time to restart from scratch and design new transport layer security protocols that will incorporate the lessons learned from the present ones, but discard the heavy baggage of old code and backward compatibility requirements. Such new protocols, which we shall call here NTLS (for New TLS) and NDTLS (for New DTLS), would provide many benefits, discussed in Section 5, and would be a stepping stone towards greater security and privacy on the Internet. The surveillance revelations make it less implausible that they will be successfully designed, standardized and deployed.

To avoid the kind of vulnerabilities that have plagued TLS, and other security protocols such as IPsec and SSH, the new protocols should be formally analyzed as they are being designed. We explain in Section 6 why this will not be easy.

# 2 Shortcomings of the Existing Transport Layer Security Protocols

Most of the following shortcomings affect both TLS and DTLS. Specific shortcomings of DTLS are mentioned in Section 2.6.

## 2.1 Latency of Handshake Roundtrips

TLS requires an initial handshake for establishment of a shared secret and for server and/or client authentication. An unabridged handshake involves two data roundtrips. Over a geostationary satellite link, signal transmission at the speed of light adds about 270 ms of latency for each roundtrip.

When a web browser retrieves a web page, the contents of the page typically include multiple items that are each retrieved from different servers using different connections. In the case of a secure page, each of those connections requires a TLS handshake. Some handshakes take place in parallel, but some take place sequentially. When the handshakes take place over geostationary satellite links, they may add several seconds to the overall latency of page retrieval.

The number of roundtrips in a given handshake can be reduced from two to one by resuming an earlier TLS session, if session parameters are still available, either in a server cache, or in the client if a TLS extension defining session tickets [14] is used. Session resumption mitigates the latency impact of the handshake, but does not eliminate it.

Other mechanisms for reducing the number of handshakes have been proposed but have not been successful:

- Fast-track [15] eliminated one roundtrip in some cases by caching server information in the client. It was not standardized or deployed.
- False Start [16] eliminated one roundtrip by letting the client start sending protected application data without waiting for the server's `Finished` message that acknowledges a successful handshake. It was deployed by Google but not standardized. It was eventually discontinued due to interoperability issues [17].
- Snap Start [18] eliminated both roundtrips in some cases by caching server information in the client and not waiting for the server. It was not standardized or deployed.

### 2.1.1 TCP Roundtrip

The roundtrips of the TLS handshake come in addition to a roundtrip needed to establish the underlying TCP connection, where the initiator of the connection (typically the TLS client) sends a SYN message, receives a SYN-ACK message, then sends an ACK message [19]. An attempt at eliminating the TCP roundtrip [20] failed as it became clear that the roundtrip is needed to eliminate denial-of-service (DOS) attacks against the receiver of the TCP connection, and against unrelated Internet hosts, based on IP address spoofing. TCP Fast Open [21], briefly discussed below in Section 4.2.1, is a new and promising attempt at eliminating the TCP roundtrip.

## 2.2 Bandwidth Consumed by Certificate Transmission

The TLS handshake requires transmission of the server certificate chain and, in the case of mutual authentication, the client certificate chain. Each certificate chain typically comprises

two or three certificates, and certificates are bulky.[1] Certificate transmission consumes a large amount of bandwidth, slowing down network traffic in bandwidth-constrained wireless networks, adding to the latency caused by roundtrips over satellite links, and draining power from battery-powered devices.

Military networks are particularly impacted by such bandwitdh consumption. The US Department of Defense recognized this problem and funded a research project that found a way of mitigating it by compressing certificates using ad-hoc pre-placed dictionaries [22]. However, the effectiveness of the compression was limited by the fact that the public key and the signature included in a certificate are not compressible. We shall see below that NTLS will eliminate the server certificate chain, and will also enable cryptographic client authentication without certificates.

## 2.3 Limited Support for Cryptographic Client Credentials

TLS provides some support for cryptographic client credentials. It allows the server to ask the client to authenticate using an X.509 public key certificate, such as may be contained in a PIV or CAC card. A TLS extension classified as informational [23] allows the use of an OpenPGP public key certificate instead of an X.509 certificate. Another extension, classified as experimental [24], allows the client to send to the server any number of attribute certificates and/or SAML assertions. However there is no way for the server to specify which client certificate and/or what set of attributes or assertions it requires. An Internet draft being considered by the IESG for adoption as a proposed standard [25] would allow the client to authenticate by sending the server an uncertified public key, but it does not specify how the client demonstrates knowledge of the corresponding private key. Support for mutual authentication using a Kerberos ticket was provided in 1999 by an IETF Proposed Standard [26] that specified 14 additional TLS ciphersuites, each combining Kerberos entity authentication during the handshake with one of the 14 combinations of data encryption and data authentication algorithms available at the time for use with TLS; but the effort was not kept up, and ciphersuites combining Kerberos entity authentication with more recent data encryption and/or data authentication algorithms have not been defined.

However TLS does not allow the client to authenticate using credentials based on any identity-based cryptosystem, nor privacy-enhancing credentials such as U-Prove tokens [27, 28], Idemix anonymous credentials [29, 30], group signature credentials [31] or selective-disclosure certificates [32]; and adding support for a new kind of client credential is a cumbersome process, requiring the creation of a large number of new ciphersuites, or the specification of an ad-hoc TLS extension.

## 2.4 Client Identity and Attributes Sent in the Clear

The client's identity certificate and/or attribute certificates and/or SAML assertions, when used, are sent in the clear, while they should be sent encrypted in the same manner as application data.

[1] A survey of Department of Defense certificates mentioned in [22] found an average size of 1077 bits in 2010, excluding larger certificates with 2048-bit public keys and signatures, which were beginning to be deployed at that time.

## 2.5 Cryptographic Vulnerabilities

Several cryptographic vulnerabilities, discussed below, have been identified in TLS over its long history. Some of them can be considered fixed. Others have been fixed, but only in versions that are not yet broadly deployed. Others have not been fixed yet as of the latest specification.

### 2.5.1 Handshake Vulnerability to Man-In-The-Middle Attack (Historic)

The handshake specified in SSL 2.0 was vulnerable to a man-in-the-middle attack that allowed the adversary to force the use of insecure ciphersuites. The vulnerability was corrected in SSL 3.0. SSL 2.0 is still supported by 25.7% of TLS servers in the above-mentioned SSL/TLS survey [12], but it is rarely used because it is either not supported or turned off by default in web browsers; so this vulnerability may be considered fixed.

### 2.5.2 RSA Timing Vulnerability (Historic)

In 1996, Paul Kocher described a timing attack that allowed the attacker to guess a secret exponent in a modular exponentiation when the modulus and base are known [33]. Such an exponentiation is performed by a TLS server with its RSA private key to decrypt the encrypted premaster secret sent by the client, in commonly used configurations. For a while it was thought that optimizations performed by TLS servers prevented timing attacks; use of the Chinese Remainder Theorem, for example, prevents the attack by replacing exponentiation modulo the public RSA modulus $n = pq$ with exponentiations modulo each secret factor $p$ and $q$. However, in 2003, Boneh and Brumley [34] used a different timing attack to recover the secret factors, and hence the RSA private key.

Both kinds of timing attacks are now avoided at the cost of a small loss of performance, using a technique originally conceived for computing blind signatures, as proposed in the Kocher paper [33]. So this vulnerability may also be considered fixed.

### 2.5.3 RSA Padding Vulnerability (Historic)

In 1998, Bleichenbacher [35] published an attack against RSA that exploited the method specified in PKCS #1 version 1.5 [36] for padding a message before encrypting it, and showed how the attack could be used to recover the premaster secret in SSL version 3, when sent encrypted with the RSA public key of the server. The attack depended on the server producing an error message when the padding was not formatted correctly, and not producing one when a ciphertext sent by the attacker happened to yield a plaintext with correct PKCS #1 padding.

In response to the attack, RSA Labs issued PKCS #1 version 2 [37], which uses Optimal Asymmetric Encryption Padding (OAEP) [38]. TLS, however, was not updated to make use of OAEP; and timing attacks against OAEP were later published [39, 40].

Today, TLS implementations avoid the attack by hiding the fact that the server has found ill-formed padding. When ill-formed padding is found, instead of sending an error message, the server continues the handshake as if the padding was correct, using a random value as the premaster secret. This is an effective countermeasure, so this vulnerability may also be considered fixed.

### 2.5.4 Chained CBC Initialization Vectors and RC4 Biases

The most commonly used TLS ciphersuites use a block cipher in cipher-block chaining (CBC) mode [41, §6.2] for encryption of application data. Up to and including TLS 1.0, in such ciphersuites, the initialization vector (IV) of a data record was the last ciphertext block of the previous record, making the protocol vulnerable to blockwise-adaptive chosen-plaintext attacks [42, §9][43][44]. Although the vulnerability was fixed in TLS 1.1 and TLS 1.2, most servers have not upgraded to TLS 1.1 or TLS 1.2 and remain vulnerable [12]. A practical exploit of this vulnerability known as the BEAST attack [13] has recently received much attention. To avoid the BEAST attack, security practitioners recommended using ciphersuites based on the RC4 stream cipher instead of a block cipher, but RC4 vulnerabilities were then discovered [45] and the use of RC4 is now discouraged [46]. As a client-side countermeasure, OpenSSL optionally inserts empty fragments (application records with no data) to randomize IVs, but this option breaks some servers.

### 2.5.5 Padding-Related Vulnerabilities

In CBC-mode ciphersuites TLS authenticates and pads application data before encrypting. Specifically, it adds a message authentication code (MAC) to the application data, then adds padding that is not covered by the MAC, then encrypts the concatenation of data, MAC and padding. Vaudenay [47] showed that by modifying the encrypted traffic in a chosen-ciphertext attack, an attacker can cause the recipient to interpret plaintext as padding and obtain information about the plaintext by detecting whether the recipient deems the padding to be well-formed. Canvel et al. [48] achieved a plaintext recovery attack against TLS 1.0 using timing analysis to detect that the MAC was not computed in cases where the padding was deemed ill-formed. TLS 1.1 and 1.2 specify that the same error message is to be sent for ill-formed padding as for an ill-formed MAC, to prevent detection (even though the error message is sent encrypted). TLS 1.1 and 1.2 also recommend computing a dummy MAC when the padding is ill-formed; however, a careful timing analysis of the MAC computation makes it possible in some cases to detect the length of its argument and hence the value of the padding, or the fact that the padding is ill-formed. This enables the attack known as *Lucky Thirteen* [49].

### 2.5.6 Compression-Related Attacks

Three recent attacks, CRIME [50], BREACH [51], and TIME [52], exploit the general fact that compression before encryption may leak information about the plaintext, as shown by Kelsey [53]. The attacker injects content into a plaintext containing a secret, and is able to recover the secret. In CRIME, the plaintext is an HTTP request that is compressed by TLS; in BREACH, the plaintext is an HTTP response encrypted by HTTP; TIME may use either kind of plaintext. After compression, the plaintext is encrypted and the attacker obtains information on the length of plaintext from the length of the ciphertext. In CRIME and BREACH the attacker observes the length of the ciphertext directly, while in TIME the attacker measures differences in ciphertext transmission time.

Since information leakage from the length of compressed plaintext is a general fact, it may be argued that the TLS protocol is not to be blamed for these attacks. However, by including a compression layer, and a CBC padding feature that the specification says is useful

for frustrating attacks based on analysis of the lengths of exchanged messages [7, 6.2.3.2], TLS does bear some responsibility. Compression is further discussed below in Section 4.9, and length-hiding in Section 4.10.

### 2.5.7 Renegotiation Vulnerability

SSL and TLS include a procedure for executing a second handshake over a TLS-protected connection, with the purpose of establishing fresh shared keys for data encryption and authentication, or performing delayed client authentication. The second handshake is called a *renegociation*. A flaw in the renegociation procedure allows an attacker to combine an initial TLS connection between the attacker and a server with a subsequent TLS connection between a victim and the server proxied via the attacker, in such a way as to make the server believe that it is communicating with the same TLS client throughout the combined connection. The TLS Renegociation Indication Extension [54] provides a countermeasure.

### 2.5.8 Global Vulnerability to the Compromise of a Single CA

In TLS the server authenticates with a public key certificate backed by a certificate chain. Any certificate authority may issue a certificate to any server, and the certificate is sent to the client by the server itself during the handshake. Hence if an attacker is able to compromise any CA, the attacker is able to impersonate any server, no matter what certificate authority the legitimate server has obtained its certificate from. CA compromises occur from time to time and cause widespread panic [55, 56].

This vulnerability was discussed at the recent NIST Workshop on Improving Trust in the Online Marketplace [57]. Many approaches to solving the problem were discussed or mentioned in workshop presentations [58], but no obvious solution emerged, according to the closing presentation [59].

## 2.6 Additional Shortcomings of DTLS

DTLS has many of the shortcomings of TLS discussed in the previous section, plus two specific shortcomings of its own.

### 2.6.1 Additional Roundtrip

DTLS requires an initial roundtrip for protection against denial-of-service attacks by an attacker who spoofs an IP address, similar to those that require the initial SYN / SYN-ACK / ACK roundtrip of TCP mentioned in Section 2.1.1. In the initial roundtrip the DTLS server sends the DTLS client a stateless "cookie," and the client returns the cookie to demonstrate that it is capable of receiving packets at its claimed IP address [60, §4.3]. This roundtrip is in addition to the one or two roundtrips of the same key exchange handshake that is used in TLS.

### 2.6.2 Amplification of Timing Attacks

In DTLS, an attacker can amplify a chosen-ciphertext timing attack by sending a ciphertext $C$ that causes a silent error $N$ times in succession immediately followed by a ciphertext

that provokes a response, then measuring the time $T$ that it takes to receive the response and compute the time $T/N$ that it takes to process $C$. This technique was first used in [61] for attacks against timing vulnerabilities in OpenSSL and GnuTLS. The vulnerabilities were easily fixed, but then the technique was used in [49] for the DTLS version of the LuckyThirteen attack.

# 3 Efforts to Address the Shortcomings

Many attempts have been made to remedy these shortcomings, but they have so far been half-hearted, piecemeal, and largely unsuccessful. We have already mentioned the proposals made to reduce the number of roundtrips, the NIST workshop where several approaches were discussed for mitigating the global vulnerability to the compromise of a single CA, the protocol extension to allow the use of X.509 attribute certificates, and the Internet draft on the use of uncertified key pairs. Proposals to encrypt the client certificate have been considered several times, but never adopted.

Each of the proposals that have been made is concerned with improving only one of the shortcomings of TLS and DTLS. Even QUIC, an entirely new transport layer security protocol being proposed by Google [62], is narrowly concerned with performance improvements, without addressing any of the other shortcomings. Today, however, the deep concern caused by the revelations of government and corporate spying on Internet users may provide the impetus to radically redesign transport layer security and address all of its current shortcomings.

# 4 Ingredients and Benefits of the New Transport Layer Security Protocols

## 4.1 Identity-Based Key Transport with DNS Support for a Zero-Roundtrip Handshake

A traditional TLS handshake without forward secrecy[2] comprises a first roundtrip where the client obtains from the server a certificate containing an encryption public key, and a second roundtrip where the client sends a secret encrypted under the public key (the *premaster secret*) and waits to receive a cryptographic acknowledgement from the server (the server's `Finished` message) before sending application data.

In NTLS, we propose to eliminate the first roundtrip, without caching server information as in Fast-track [15] or Snap Start [18], by using Hierarchical Identity-Based Encryption (HIBE) [63, 64] with assistance from the Domain Name System (DNS) [65, 66], without necessarily requiring the security provided by the Domain Name System Security Extensions (DNSSEC) [67, 68]. Several identity-based protocols have been proposed for key agreement, but they are not concerned with eliminating roundtrips. Identity-based encryption is being used commercially to provide encrypted email [69]. The proposed use of identity-based encryption to implement NTLS will broaden the practical applications of the technology.

[2] *Forward secrecy*, also known as *Perfect Forward Secrecy (PFS)*, refers to the immunity of session keys against disclosure of long term keys that is achieved by an ephemeral Diffie-Hellman key exchange.

We propose to slightly generalize the HIBE paradigm to allow for multiple root Private Key Generators (PKGs), analogous to multiple root certificate authorities (CAs). Hierarchies of PKGs will have any number of levels. In a two-level hierarchy a root PKG will issue a private key to a subordinate PKG, which will issue a private key to a TLS server. The root PKG will have cryptographic *public parameters*. The TLS server will have an *identity chain* consisting of its identity, followed by the identity of the subordinate PKG and the identity of the root PKG.[3] In a hierarchy with more than two levels there will be intermediate PKGs between the PKG that issues its private key to the server and the root PKG, each intermediate PKG issuing a private key to the previous PKG in the chain and receiving its own private key from the next PKG in the chain. The public key of the server will consist of the identity chain and the public parameters of the root PKG.[4]

Each NTLS client will store the public parameters of the root PKGs that it trusts (as TLS clients store today the certificates of the root CAs that they trust). The identity of a server will be its hostname in the DNS. PKG identities will not necessarily be DNS names, but the *identity tail* of a server, consisting of its identity chain minus the identity of the server itself, will be stored in the DNS.[5] To establish a secure connection to an NTLS server, a client will obtain the identity tail of the server at the same time as the server's IP address.[6] The client will use the identity of the root PKG found in the identity tail to find the public parameters of that root PKG in the client's store of trusted public parameters. It will then add the hostname of the server to the identity tail to form the identity chain, and use the public parameters and the identity chain as the public key of the server, thus avoiding the need to retrieve a certificate chain and saving one roundtrip. The client will then encrypt a high-entropy random secret under the server's public key and send it to the server, thus establishing an *initial shared secret*.

To eliminate the second roundtrip, the client will start sending application data to the server in the very first message, the same message that carries to the server the initial shared secret. This *first-message data* will be encrypted and authenticated with *initial shared keys* derived from the initial shared secret.

Since the client sends the first-message data to the server before the server has authenticated, it may well be sending it to an adversary. This is justified by the fact that the recipient will only be able to decrypt the data if it is the authentic server. Google's False Start [16] was also based on the idea of sending data without waiting for the server to confirm its identity. It failed because False Start deviated from the formal specification of the TLS protocol and Google did not propose an upgrade of the protocol [17]. Since NTLS will be a brand new protocol, it will not face the same difficulty.

Since the server does not contribute randomness to the generation of the initial shared keys, it will be possible for an attacker to replay the first message. However, application servers should be able to cope with receiving multiple copies of the first message data; this is the case in particular for web servers, since users often repeat an unsuccessful request

[3] In typical theoretical treatments of HIBE, such as [64], the root PKG does not have an identity because there is only one root PKG and only one set of public parameters. In our context, each root PKG must have an identity, which the client uses to locate its particular set of public parameters.

[4] Notice that subordinate PKGs do not have associated public parameters.

[5] This will require a DNS extension. Extending the DNS to store data for various protocols is common practice. See, for example, [70, 71, 72].

[6] The client will issue a DNS A query. The response to the query will carry the IP address in the Answer Section, and the identity tail in the Additional Section.

to access a site. Applications that cannot tolerate first message replay will send an empty first message, at the cost of a roundtrip. To prevent an entire session from being replayed, the server will generate a high-entropy random secret and send it in the second message, encrypted under the initial shared secrets. Client and server will then combine this server secret with the initial shared secret chosen by the client to generate a *subsequent shared secret*, from which they will derive *subsequent shared keys* for encrypting and authenticating application data starting with the second message. An alternative method of generating the subsequent shared keys with forward secrecy is described below in Section 4.6.

### 4.1.1 Complementary Use of TCP Fast Open

As we said above, TCP Fast Open [21] (briefly discussed below in Section 4.2.1) is a promising attempt at dispensing with the TCP roundtrip. If TCP Fast Open is successfully deployed it will be a perfect complement to NTLS. Together, TCP Fast Open and NTLS would completely eliminate roundtrip latency of secure connection establishment over satellite links.

## 4.2 Elimination of the DTLS Stateless Cookie Handshake

In Section 2.6.1 we saw that DTLS requires an additional roundtrip for preventing DOS attacks based on IP address spoofing, in which the server sends a stateless cookie and the client returns it to demonstrate that it is capable of receiving packets at its claimed IP address.

In NDTLS we propose to avoid this cookie roundtrip as follows. When a client establishes a NDTLS session with a server with which the client has not interacted before, or with which the client has interacted but forgotten the interaction, an initial roundtrip will take place as in DTLS, using a cookie of a first type, which is sent in the clear. Then, upon successful establishment of an NDTLS session, a cookie of a second type will be sent from the server to the client, encrypted by the NDTLS session. When the client establishes a subsequent session with the server, it will send the previously obtained cookie of the second type, encrypted under the server's HIBE public key, to avoid the cookie roundtrip.

### 4.2.1 Comparison with TCP Fast Open

TCP Fast Open [21] avoids the TCP roundtrip by caching a cookie, and keeping a counter of pending connections, which triggers a fallback to using the ordinary TCP handshake when the counter exceeds a threshold during a DOS attack. In NDTLS such a counter is not needed because the cookie of the second type is sent encrypted and is therefore not available to a DOS attacker.

### 4.2.2 Comparison with QUIC

QUIC [62] is a protocol that multiplexes data streams transported over UDP, with protection of UDP datagrams by ad-hoc cryptography similar to TLS [73]. QUIC uses "source address tokens" that play the same role as the cookies mentioned above in connection with DTLS, TCP Fast Open, and NDTLS. The tokens can be sniffed by a DOS attacker. As mitigation, instead of using a counter as in TCP Fast Open, QUIC relies on reducing the lifetime of the tokens. NDTLS does not need to reduce the lifetime of cookies of the second type, because

those cookies are sent encrypted and cannot be sniffed by a DOS attacker who is spoofing an IP address.

We believe that the design goals of QUIC, which include zero-roundtrip session establishment, could be more consistently achieved, together with many additional security, privacy and usability benefits, by replacing the ad-hoc cryptography of QUIC with NDTLS, or by using the Stream Control Transmission Protocol (SCTP) over NDTLS.

## 4.3 Eliminating Certificate Transmission

As we have just seen, the client will not need to retrieve a certificate chain from the server. Instead it will retrieve an identity tail from the DNS at the same time as the server's IP address. The identity tail consists of the names of the root PKG and any subordinate PKGs, and names need not be more than a few bytes long; so transmission of the identity tail will consume negligible bandwidth.

In NDTLS, not having to transmit the server certificate chain will reduce or eliminate the need to fragment datagrams. Fragmentation degrades performance of UDP-based protocols. Avoiding fragmentation was an explicit design goal for the DTLS record layer [60, §4.1], but one that could not be met by the handshake portion of the protocol [60, §4.3], precisely because of the need to transmit the server certificate chain. QUIC tries to avoid fragmentation by sending hashes of certificates in the certificate chain, hoping that the client will already know the certificates, at the cost of increasing the certificate storage requirements of the client.

If the client uses a certificate for authentication, it must send its certificate chain to the server. However, as discussed below, NTLS (and NDTLS) will allow for the use of many different kinds of credentials. If the user uses as a credential a key pair pertaining to an identity-based signature scheme, where the public key is the user's identity, there will be no need to send a certificate, and certificate transmission will be avoided entirely.

## 4.4 Mitigation and Future Elimination of the Global Vulnerability to a Compromise of a Single Trust Provider

A PKG used in NTLS plays the same role that a CA plays in TLS. Let us generically refer to PKGs and CAs as Trust Providers (TPs). We saw in Section 2.5.8 that, in TLS, the client retrieves the server's certificate chain from the server itself, and therefore an attacker can impersonate any server by compromising a single TP. In NTLS, on the other hand, the client retrieves the identity tail from the DNS. Therefore an attacker who compromises a different TP than the one used by the server cannot use a credential issued by the compromised TP unless the attacker also tampers with the DNS response obtained by the client. Since DNSSEC is not yet universally deployed, NTLS also has a global vulnerability to a single TP compromise. However the vulnerability is mitigated by the need to tamper with the DNS, and will be eliminated once DNSSEC is universally deployed.

### 4.4.1 Comparison with DANE

The DANE protocol (DNS-Based Authentication of Named Entities [72]) also relies on the DNS to address the single-TP compromise vulnerability. DANE stores either the server

certificate or a hash of the certificate in the DNS, relying on DNSSEC for security. However, DANE does not modify TLS, and thus the client retrieves the certifiate from the server during the handshake. DANE requires the client to compare the certificate retrieved from the server with the certificate or hash obtained from the DNS. Thus DANE does not eliminate roundtrips, and increases bandwidth consumption instead of reducing it.

## 4.5 Short-Term Server Credential and Emergency Revocation

A difficulty with identity-based cryptography is the fact that a credential based on a long-term identity cannot expire or be revoked. In NTLS we propose to use two credential-termination mechanisms:

1. The identity of the server used as a component of the server's public key will be augmented by a short-term expiration time. The expiration time will be stored in the DNS and retrieved by the client together with the identity tail, when the client looks up the server's IP address. The server will obtain a new private key for each expiration time from its private key generator.

2. The identity of the server will be further augmented by a revocation counter, which will also be obtained from the DNS along with the identity tail and the expiration time. The revocation counter will be incremented whenever the server's credential needs to be revoked before its expiration time. A new private key will then be obtained by the server from the PKG.

The first mechanism does not rely on DNS security. The second mechanism will be fully effective once DNSSEC is universally implemented, and partially effective in the meantime.

## 4.6 Optional Forward Secrecy After First Message

Today, TLS provides optional forward secrecy using ephemeral Diffie-Hellman key agreement. In the first roundtrip the server sends a certificate binding the identity of the server to a long-term public key pertaining to a signature cryptosystem (either RSA, DSA or ECDSA), and ephemeral Diffie-Hellman parameters signed with the corresponding long term private key. The parameters include the Diffie-Hellman modulus and generator, chosen by the server, and the server's ephemeral Diffie-Hellman public key. In the second roundtrip the client sends its ephemeral Diffie-Hellman public key. Then client and server use the resulting Diffie-Hellman shared secret as the premaster secret.

While it is not possible to provide forward secrecy for the very first message, NTLS will provide optional forward secrecy starting with the second message.[7] In many cases forward secrecy is not needed for the first message, e.g. because first-message data is not even confidential. When needed, the client will be able to achieve complete forward secrecy by sending no data with the first message, at the cost of a roundtrip.

To provide forward secrecy starting with the second message we propose to use ephemeral Diffie-Hellman key agreement as follows. The modulus and the generator will be chosen by the client rather than the server, and sent to the server together with the client's ephemeral

[7] After writing the first version of this paper, we have found out that delayed forward secrecy is used in QUIC [62].

Diffie-Hellman public key in the first message (in addition to the other contents of the first message described above in Section 4.1). Then the server will send its ephemeral Diffie-Hellman public key in the second message and both client and server will compute the Diffie-Hellman shared secret. The Diffie-Hellman parameters exchanged by the client and server will be encrypted and authenticated by the initial shared keys. The client will trust that the ephemeral public key that it receives belongs to the server because the initial shared keys are derived from the initial shared secret, which the client sent to the server encrypted under the server's identity-based public key. The "subsequent shared keys" for encrypting and authenticating data starting with the second message will be derived from the Diffie-Hellman shared secret.

The forward secrecy feature will be optional. Either party to the protocol (client or server) will be able to offer the feature by sending its Diffie-Hellman parameters and may choose to either close the connection or proceed without the feature if the other party does not offer it.

## 4.7 Generic Cryptographic Client Authentication Independent of the Handshake

A mechanism will be provided for specifying and registering presentation protocols for a variety of cryptographic client credentials besides public-key certificates, without having to specify extensions of NTLS. It will be possible to specify presentation protocols for key pairs pertaining to identity-based signature schemes [74] where the public key is the user's identity. It will also be possible to specify presentation protocols for privacy-enhancing credentials such as group signature credentials [31], U-Prove tokens [27, 28], Idemix anonymous credentials [29, 30], and selective-disclosure certificates [32]. Last but not least, it will also be possible to use a simple uncertified key pair as a client credential, gaining the benefits mentioned below in Section 4.7.1.

Credential presentation will be independent from the handshake. The server will be able to ask for a set of client credentials at any time and the client will be able to present the credentials over an existing NTLS connection without having to execute a new handshake. All credential presentations will benefit from the confidentiality provided by the connection and will derive the challenge typically needed to demonstrate possession of a credential from the NTLS shared secret.

### 4.7.1 Benefits of Uncertified Key Pairs

Although uncertified key pairs are not usually classified as privacy-enhancing credentials, they can be used as a replacement for ordinary passwords, greatly increasing security without sacrificing privacy by involving a third-party credential issuer. Uncertified key pairs are also useful for authentication of autonomous devices in the Internet of Things.

## 4.8 Dispensing with Session Cookies

The dissociation of client authentication from the handshake will make it possible to dispense with session cookies. A login session will be an authenticated portion of an NTLS

session, starting when the client authenticates, and ending when the client sends an end-of-authenticated-session message to the server.

## 4.9 No Compression

Because compression before encryption may leak information that defeats the purpose of the encryption [53], the new protocol will not include a compression phase. Compression may be necessary for performance, but we believe that it is best left to the application layer. In fact, we have conceived a compression method that an application could use to prevent the leakage of secrets, using application-specification knowledge of what secrets need to be protected and where they are located within the application data. Different portions of the application data would be marked as belonging to different "compression contexts," and would thereby be compressed separately using different compression dictionaries. The application would place secrets and data potentially controlled by an adversary in different compression contexts. The details are out of the scope of this paper.

## 4.10 No Padding

As we have seen above, padding has been the source of several TLS vulnerabilities, some of which are yet to be remedied. So padding is best avoided if possible.

In TLS, padding is used for two purposes: to fill plaintext blocks in CBC mode, and to hide the length of the plaintext. Length hiding is a traffic analysis countermeasure whose purpose is to prevent an adversary from distinguishing a plaintext among a collection of possible plaintexts, based on the length of the ciphertext. But length hiding is most effective when based on knowledge of the collection of possible plaintexts, and such knowledge is application-specific.[8] Therefore padding for the purpose of length hiding should be performed by the application layer rather than by the transport security layer; we agree with Shoup [76, §2.2.2] that "it is in general up to the application using the cryptosystem to ensure that the length of a message does not reveal sensitive information."

As to padding for the purpose of filling plaintext blocks, it can be avoided by using traditional block cipher modes other than CBC, such as CFB[9], or preferably OFB[10], which allows for precomputation of block cipher operations, thus potentially providing better performance. It can also be avoided by using an AEAD cipher mode such as AES-GCM.

## 4.11 Encryption before Authentication

For a variety of reasons, it is generally considered that the most secure way of combining encryption and authentication of application data is to first encrypt the data and then authenticate the ciphertext [42, §4][77][78]. We have seen that TLS does the opposite.

[8] Paterson et al. [75] assert that TLS provides "length-hiding authenticated encryption (LHAE)". But their LHAE notion involves a weak adversary that has less information than is available to real world adversaries (such as those the carry out the CRIME or BREACH attacks). In particular the LHAE adversary cannot determine the minimal number of ciphertext blocks that a plaintext can be encrypted into, whereas a real-world adversary usually can.

[9] Cipher Feedback Mode [41, §6.3].

[10] Output Feedback Mode [41, §6.4].

Most of the attacks against TLS mentioned above in Section 2.5, including those related to chained initialization vectors (Section 2.5.4) and those related to padding vulnerabilities (Section 2.5.5) are chosen ciphertext attacks that would have been thwarted by a MAC applied to the ciphertext.

An Internet draft proposing a TLS extenstion that would change the order of encryption and authentication of application data [79] has been discussed on the mailing list of the TLS working group [80] but has not been classified as an officially active draft [81].

An alternative to using separate encryption and MAC algorithms is to use a block cipher such as AES in a mode of operation that simultaneously provides encryption and authentication, such as Galois Counter Mode (GCM). Such use of a block cipher is referred to as Authenticated Encryption with Associated Data (AEAD) algorithm. The ability to use AEAD was added to TLS in TLS 1.2 [7, §6.2.3.3]. Specific AEAD ciphersuites were added later [82, 83].

In NTLS we propose to allow for both encryption followed by authentication and AEAD.

# 5 Summary of Benefits of the Proposed Transport Layer Security Protocols

The new protocols will provide the following benefits:

- They will increase the performance of secure connections over satellite and radio links, and improve the battery life of battery-powered devices.
- They will facilitate the transition from the current World Wide Web where most connections are not protected by transport layer security, to a future web where all connections are protected, by making the performance of secure connections acceptable even when users access the web from ships, airplanes, or remote areas using satellite or cellular networks.
- They will strengthen transport layer security by eliminating current vulnerabilities and putting it on a sounder cryptographic footing.
- They will facilitate an increase in the use of cryptographic credentials for client authentication, and an ensuant reduction in the use of passwords.
- They will provide confidentiality and unlinkability of cryptographic client credentials by eliminating the current TLS practice of sending client certificates in the clear, and by enabling the use of anonymous credentials. This will increase privacy on the Internet and mitigate traffic analysis.
- They will allow the use of uncertified key pairs, both as privacy-preserving replacements of ordinary passwords, and as credentials of autonomous devices in the Internet of Things.

# 6 The Need to Prove Security

To avoid repeating the history of vulnerabilities and surprise attacks that have plagued TLS and DTLS, it will be highly desirable to formally prove the security of the new protocols before deploying them.

Some of the above ingredients, such as the overlap of key exchange with application data transmission, and the use of two different key agreements in succession (key transport followed by ephemeral Diffie-Hellman), will cause the proof of security to be challenging. Furthermore, to be meaningful, the proof of security should include assurance of protection against timing attacks, and validation of the NDTLS countermeasures against certain DOS attacks.

On the other hand the proof of security will be facilitated by some of the features and design choices of the new protocols, viz. the fact that the new protocols will encrypt application data before authenticating it, and will leave compression and length hiding to the application layer.

The proof of security that will build upon two decades of research work on provable security of communication protocols and make use of techniques that have emerged from that work, such as the use of sequences of games to structure complex proofs [84].

It will build more particularly upon the work that has been done over the last few years on analyzing TLS.

Recently, Jager *et al.* [85] managed to overcome a difficulty that had prevented earlier researchers from proving security while accurately modeling the `Finished` messages that conclude the handshake. Traditionally, security of the key exchange was formulated by requiring that an adversary not be able to distinguish between the keys produced by the handshake and randomly generated keys; but because the Finished messages include known plaintext encrypted by those keys, the adversary can use them to distinguish the real keys from the random ones. To overcome this obstable Jager *et al.* introduced the notion of Authenticated and Confidential Channel Establishment (ACCE), a notion of security that can be applied to the entire TLS protocol, without relying on a separate notion of security for the key exchange. They were then able to prove the security of TLS for ciphersuites based on ephemeral Diffie-Hellman with mutual client and server authentication, under assumptions about the security of the TLS Record Layer as it transmits and protects application data.

Then Krawczyk *et al.* [86] generalized the approach using the concept of a Key-Encapsulation Module (KEM) used by Jonsson and Kaliski [87], and were able to prove security for ciphersuites based on RSA key transport, and for ciphersuites where only the server authenticates.

While ACCE overcomes the problem presented by the encrypted `Finished` messages, the proofs in [85, 86] still rely on TLS having two distinct phases: the key exchange phase, including the `Finished` messages, and the subsequent secure channel phase in which application data is transmitted. By contrast, in NTLS and NDTLS there will be complete overlap between key exchange and data transmission; therefore other approaches may be more suitable for the new protocols. One promising approach is that of Bruzska *et al.* [88], which provides the benefit of composability without relying on a temporal divide between two phases.

# 7 Conclusion

We have discussed the security, privacy and usability shortcomings of the current transport layer security protocols, TLS and DTLS. We have argued that awareness of the breakdown of privacy and security on the Internet provides an opportunity to redesign transport layer security from scratch, and listed several ingredients that could be used in such a redesign, including the use of multi-root, hierarchical identity-based encryption for key exchange, and the dissociation of client authentication from the key exchange to allow for an open-ended variety of client credentials and for the implementation of login sessions without cookies. The redesigned protocols will increase performance over satellite and radio links, enabling ubiquitous use of secure connections; they will strengthen security by eliminating vulnerabilities and putting the protocols on a sounder cryptographic footing; and they will increase privacy by facilitating the use of uncertificated key pairs and privacy-enhancing credentials for client authentication.

# References


[1] Jeff Larson. NIST to review standards after cryptographers cry foul over NSA meddling. Network World. November 5, 2013. http://www.networkworld.com/news/2013/110513-nist-nsa-275641.html.

[2] Alex Wilhelm. NSA Reportedly Paid A Security Firm Millions To Ship Deliberately Flawed Encryption Technology. December 21, 2013. http://techcrunch.com/2013/12/20/nsa-reportedly-paid-a-security-firm-millions-to-ship-deliberately-flawed-encryption-technology/.

[3] Yasha Levine. Googles for-profit surveillance problem. December 16, 2013. http://pando.com/2013/12/16/googles-for-profit-surveillance-problem/.

[4] US Department of Defense. Architecture for Enterprise Anonymization. Air Force SBIR soliciation 2014.1, topic AF141-058. http://www.zyn.com/sbir/sbres/sbir/dod/af/af141058.htm.

[5] T. Dierks and C. Allen. The TLS Protocol Version 1.0, January 1999. http://tools.ietf.org/html/rfc2246.

[6] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1, April 2006. http://tools.ietf.org/html/rfc4346.

[7] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2, August 2008. http://tools.ietf.org/html/rfc5246.

[8] E. Rescorla and N. Modadugu. Datagram Transport Layer Security, April 2006. http://tools.ietf.org/html/rfc4347.

[9] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2, January 2012. http://tools.ietf.org/html/rfc6347.

[10] National Security Agency. Global Information Grid. http://www.nsa.gov/ia/programs/global_information_grid/index.shtml.

[11] Eric Rescorla. *SSL and TLS—Designing and Building Secure Systems*. Addison-Wesley, 2001.

[12] Trustworthy Internet Movement. SSL Pulse—Survey of the SSL Implementation of the Most Popular Web Sites. November 2, 2013. Retrieved from https://www.trustworthyinternet.org/ssl-pulse/.

[13] Thai Duong and Juliano Rizzo. Here Come the ⊕ Ninjas. May 13, 2011. Available from http://blog.tempest.com.br/static/attachments/marco-carnut/driblando-ataque-beast-com-pasme-rc4/ssl_jun21.pdf.

[14] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077. http://tools.ietf.org/html/rfc5077.

[15] Hovav Shacham, Dan Boneh, and Eric Rescorla. Client-Side Caching for TLS. *ACM Transactions on Information and System Security*, 7(4):553–575, November 2004.

[16] A. Langley, N. Modadugu, and B. Moeller. Transport Layer Security (TLS) False Start. draft-bmoeller-tls-falsestart-00, June 2, 2010. https://tools.ietf.org/html/draft-bmoeller-tls-falsestart-00.

[17] Adam Langley. False Start's Failure (11 Apr 2012). https://www.imperialviolet.org/2012/04/11/falsestart.html.

[18] A. Langley. Transport Layer Security (TLS) Snap Start. May 2010. https://www.imperialviolet.org/binary/draft-agl-tls-snapstart-00.html.

[19] ISI-USC. Transmission Control Protocol. Internet Standard. September 1981. http://tools.ietf.org/html/rfc793.

[20] R. Braden. T/TCP – TCP Extensions for Transactions. July 1994. http://tools.ietf.org/html/rfc1644.

[21] Sivasankar Radhakrishnan, Yuchung Cheng, Jerry Chu, Arvind Jain, and Barath Raghavan. TCP Fast Open. In *Proceedings of the 7th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2011. http://research.google.com/pubs/pub37517.html.

[22] Efficient Transmission of DoD PKI Certificates in Tactical Networks. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA534405.

[23] N. Mavrogiannopoulos and D. Gilmor. Using OpenPGP Keys for Transport Layer Security (TLS) Authentication. February 2011. http://tools.ietf.org/html/rfc6091.

[24] M. Brown and R. Housley. Transport Layer Security (TLS) Authorization Extensions. May 2010. http://tools.ietf.org/html/rfc5878.

[25] P. Wouters, H. Tschofenig, J. Gilmore, S. Weiler, and T. Kivinen. Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). October 19, 2013. http://tools.ietf.org/html/draft-ietf-tls-oob-pubkey-10.

[26] A. Medvinsky and M. Hur. Addition of Kerberos Cipher Suites to Transport Layer Security (TLS). October 1999. http://tools.ietf.org/html/rfc2712.

[27] Microsoft Corporation. U-Prove Home Page. http://www.microsoft.com/u-prove.

[28] Christian Paquin. U-Prove Cryptographic Specification V1.1 Draft Revision 1, February 2011. There is no http URL for this document, but it can be downloaded by following links from http://www.microsoft.com/u-prove.

[29] J. Camenisch, P. Bichsel, and T. Gross. Idemix Blog. http://idemix.wordpress.com/.

[30] IBM Research—Zurich. Specification of the Identity Mixer Cryptographic Library, Version 2.3.4, February 10, 2012. Downloadable from https://prime.inf.tu-dresden.de/idemix/.

[31] Kazue Sako. Group signatures. Presentation at NIST Meeting on Privacy-Enhancing Cryptography, December 2011. http://csrc.nist.gov/groups/ST/PEC2011/presentations2011/sako.pdf.

[32] Francisco Corella. Method and apparatus for providing field confidentiality in digital certificates, October 2004. US Patent 6,802,002.

[33] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. http://www.fit.vutbr.cz/~cvrcek/cards/timingattack/timingat.htm.en.

[34] D. Boneh and D. Brumley. Remote timing attacks are practical. Usenix Security Symposium, 2003. http://crypto.stanford.edu/~dabo/abstracts/ssl-timing.html.

[35] Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. CRYPTO 1998. http://www.bell-labs.com/user/bleichen/papers/noisy.ps.

[36] B. Kaliski. PKCS #1: RSA Encryption Version 1.5. November 1993. Republished as RFC 2312. http://tools.ietf.org/html/rfc2313.

[37] B. Kaliski and J. Staddon. PKCS #1: RSA Cryptography Specifications Version 2.0. September 1998. Republished as RFC 2437. http://tools.ietf.org/html/rfc2437.

[38] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption — How to Encrypt with RSA. Eurocrypt 94. LNCS 950. Full paper at http://cseweb.ucsd.edu/users/mihir/papers/oae.pdf.

[39] James Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. CRYPTO 2001.

[40] Falko Strenzke. Manger's Attack Revisited. In *Proceedings of the 12th International Conference on Information and Communications Security*, ICICS'10, pages 31–45, 2010.

[41] Morris Dworkin. Recommendation for Block Cipher Modes of Operation - Methods and Techniques. NIST Special Publication 800-38A. December 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[42] P. Rogaway. Problems with Proposed IP Cryptography, 1995. http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt.

[43] Gregory Bard. Vulnerability of SSL to Chosen-Plaintext Attack, 2004. http://eprint.iacr.org/2004/111.pdf.

[44] Gregory Bard. A Challenging but Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL, 2006. http://eprint.iacr.org/2006/136.pdf.

[45] Nadhem AlFardan, Dan Bernstein, Kenny Paterson, Bertram Poettering, and Jacob Schuldt. On the Security of RC4 in TLS and WPA. 13th March 2013 (updated 8th July and 28th August 2013). http://www.isg.rhul.ac.uk/tls/.

[46] Dennis Fisher. Microsoft Warns Customers Away From SHA-1 and RC4. November 12, 2013. http://threatpost.com/microsoft-warns-customers-away-from-sha-1-and-rc4/102902.

[47] Serge Vaudenay. Security Flaws Induced by CBC Padding—Applications to SSL, IPSEC, WTLS.... In *Proceedings of In Advances in Cryptology—EUROCRYPT'02*, pages 534–546. Springer-Verlag, 2002. http://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf.

[48] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password Interception in a SSL/TLS Channel. In *CRYPTO*, pages 583–599, 2003. http://www.iacr.org/archive/crypto2003/27290581/27290581.pdf.

[49] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. 27th February 2013, http://www.isg.rhul.ac.uk/tls/TLStiming.pdf.

[50] Juliano Rizzo and Thai Duong. The CRIME attack. Presentation at the 9th Ekoparty Security Conference, Buenos Aires, September 2012. Available at https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit?pli=1#slide=id.g1d134dff_1_222.

[51] Yoel Gluck, Neal Harris, and Angelo Prado. BREACH: Reviving THE CRIME Attack. http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf.

[52] Barry Shteiman, Tal Be'ery, and Amichai Shulman. A Perfect CRIME? Only TIME will tell. March 19, 2013. http://blog.imperva.com/2013/03/a-perfect-crime-only-time-will-tell.html.

[53] John Kelsey. Compression and Information Leakage of Plaintext. http://www.iacr.org/cryptodb/archive/2002/FSE/3091/3091.pdf.

[54] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. http://tools.ietf.org/html/rfc5746.

[55] Phillip Hallam-Baker. The Recent RA Compromise. March 23, 2011. http://blogs.comodo.com/it-security/data-security/the-recent-ra-compromise/.

[56] Aart Jochem. Keynote — Lessons learned from the DigiNotar case. Presentation at the NIST Workshop on Improving Trust in the Online Marketplace. April 11, 2013. http://csrc.nist.gov/groups/ST/ca-workshop-2013/presentations/Jochem_ca-workshop2013.pdf.

[57] NIST. Workshop on Improving Trust in the Online Marketplace. April 10–11, 2013. http://www.nist.gov/itl/csd/ct/ca_workshop.cfm.

[58] NIST. Agenda of the Workshop on Improving Trust in the Online Marketplace. April 10–11, 2013. http://www.nist.gov/itl/csd/ct/ca-workshop-agenda2013.cfm.

[59] Tim Polk. Building Consensus. Closing presentation at the NIST Workshop on Improving Trust in the Online Marketplace. April 11, 2013. http://csrc.nist.gov/groups/ST/ca-workshop-2013/presentations/Polk_ca-workshop2013.pdf.

[60] Nagendra Modadugu and Eric Rescorla. The Design and Implementation of Datagram TLS. In *In Proceedings NDSS*, 2004. http://crypto.stanford.edu/~nagendra/papers/dtls.pdf.

[61] Nadhem J. AlFardan and Kenneth G. Paterson. Plaintext-Recovery Attacks Against Datagram TLS. NDSS Symposium 2012. http://www.internetsociety.org/plain-text-recovery-attacks-against-datagram-tls.

[62] Jim Roskind. QUIC: Multiplexed Stream Transport over UDP. First draft: April 2012. Latest revision: December 2013. https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit.

[63] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002. http://theory.stanford.edu/~horwitz/pubs/hibe.pdf.

[64] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *EUROCRYPT*, pages 223–238, 2004. Full version available at http://crypto.stanford.edu/~xb//eurocrypt04b/bbibe.pdf.

[65] P. Mockapetris. DOMAIN NAMES - CONCEPTS AND FACILITIES. November 1987. http://tools.ietf.org/html/rfc1034.

[66] P. Mockapetris. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. November 1987. http://tools.ietf.org/html/rfc1034.

[67] R. Arends, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. March 2005. http://tools.ietf.org/html/rfc4033.

[68] S. Weiler and D. Blacka, Eds. Clarifications and Implementation Notes for DNS Security (DNSSEC). February 2013. http://tools.ietf.org/html/rfc6840.

[69] Voltage Security, Inc. Voltage SecureMail—Easy-to-use global scale email encryption inside & outside enterprise. http://www.voltage.com/products/securemail/.

[70] M. Richardson. A Method for Storing IPsec Keying Material in DNS. February 2005. http://tools.ietf.org/html/rfc4025.

[71] J. Schlyter and W. Griffing. Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. January 2006. http://tools.ietf.org/html/rfc4255.

[72] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. August 2012. http://tools.ietf.org/html/rfc6698.

[73] Adam Langley and Wan-Teh Chang. QUIC Crypto. June 20, 2013. https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/edit?pli=1.

[74] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *International Cryptology Conference*, pages 47–53, 1984. http://link.springer.com/content/pdf/10.1007%2F3-540-39568-7_5.pdf.

[75] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. In *ASIACRYPT*, pages 372–389, 2011.

[76] Victor Shoup. A Proposal for an ISO Standard for Public Key Encryption (version 2.1). December 20, 2001. http://www.shoup.net/papers/iso-2_1.pdf.

[77] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: how Secure is SSL?). In *CRYPTO*, pages 310–331. Springer-Verlag, 2001. http://eprint.iacr.org/2001/045.ps.gz.

[78] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008. http://cseweb.ucsd.edu/users/mihir/papers/oem.pdf.

[79] P. Gutmann. Encrypt-then-MAC for TLS. February 8, 2013. Latest revision October 4, 2013. http://tools.ietf.org/html/draft-gutmann-tls-encrypt-then-mac-04.

[80] TLS Working Group. Mailing List Archive. http://www.ietf.org/mail-archive/web/tls/.

[81] TLS Working Group Documents. http://datatracker.ietf.org/wg/tls/.

[82] J. Salowey, A. Choudhury, and D. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS. August 2008. http://tools.ietf.org/html/rfc5288.

[83] D. McGrew and D. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). July 2012. http://tools.ietf.org/html/rfc6655.

[84] Victor Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. First public version: November 30, 2004. Revised: January 18, 2006. http://shoup.net/papers/games.pdf.

[85] Jager, Tibor and Kohlar, Florian and Schäge, Sven and Schwenk, Jörg. On the Security of TLS-DHE in the Standard Model. In *CRYPTO 2012*, pages 273–293, 2012. Full version at eprint.iacr.org/2011/219.pdf.

[86] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the Security of the TLS Protocol: A Systematic Analysis. In *CRYPTO 2013*, pages 429–448, 2013. Full version at http://eprint.iacr.org/2013/339.pdf.

[87] J. Jonsson and B. S. Kaliski Jr. On the security of RSA encryption in TLS. CRYPTO 2002. LNCS 2442. http://link.springer.com/content/pdf/10.1007%2F3-540-45708-9_9.pdf.

[88] C. Brzuska, M. Fischlin, N.P. Smart, B. Warinschi, and S.C. Williams. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, 2013. Full version at http://eprint.iacr.org/2012/242.pdf.