Reactive and Proactive Standardisation of TLS

Kenneth G. Paterson and Thyla van der Merwe

Information Security Group,
Royal Holloway, University of London

Abstract. In the development of TLS 1.3, the IETF TLS Working Group has adopted an "analysis-prior-to-deployment" design philosophy. This is in sharp contrast to all previous versions of the protocol. We present an account of the TLS standardisation narrative, examining the differences between the reactive standardisation process for TLS 1.2 and below, and the more proactive standardisation process for TLS 1.3. We explore the possible factors that have contributed to the shift in the TLS WG's design mindset, considering the protocol analysis tools available, the levels of academic involvement and the incentives governing relevant stakeholders at the time of standardisation. In an attempt to place TLS within the broader realm of standardisation, we perform a comparative analysis of standardisation models and discuss the standardisation of TLS within this context.

Keywords: security, standardisation, TLS

1 Introduction

The Transport Layer Security (TLS) protocol is used by millions, if not billions, of users on a daily basis and is the *de facto* standard when it comes to securing communications on the World Wide Web. The protocol was initially developed by Netscape Communications under the name Secure Sockets Layer (SSL) and then officially came under the auspices of the Internet Engineering Task Force (IETF) in the mid 1990s, eventually leading to the release of TLS 1.0 [32] in 1999. Subsequent versions were released in 2006 (TLS 1.1, [33]) and 2008 (TLS 1.2, [34]). Since then, TLS has received increasing amounts of attention from the security research community. Dozens of research papers on TLS have been published, containing both positive and negative results for the protocol. What began as a trickle of papers has, in the last five years, become a flood. Arguably, the major triggers for this skyrocketing in interest from the research community were the TLS Renegotiation flaw of Ray and Rex in 2009 and the BEAST and CRIME attacks in 2011 and 2012.

The many weaknesses identified in TLS 1.2 and below, as well as increasing pressure to improve the protocol's efficiency (by reducing its latency in establishing an initial secure connection) prompted the IETF to start drafting the next version of the protocol, TLS 1.3, in the Spring of 2014. Unlike the development process employed for earlier versions, the TLS WG has adopted an "analysis-prior-to-deployment" design philosophy, making a concerted effort to engage the research

community in an attempt to catch and remedy weaknesses before the protocol is finalised.

Given the critical nature of TLS, the recent shift in the IETF's design methodology for TLS 1.3, and TLS 1.3 now reaching the beginning of the end of the standardisation process, we think it pertinent that the TLS standardisation story be told. Prior to the standardisation of TLS 1.3, the TLS WG conformed to a reactive standardisation process: attacks would be announced and the WG would respond to these attacks by either updating the next version of the protocol or by releasing patches for the TLS standard. A number of factors contributed to the adoption of such a standardisation process. As we argue in the sequel, protocol analysis tools were not mature enough at the time of the design, the research community's involvement in the standardisation process was minimal, and until the first wave of attacks in 2009-2012, attacks on TLS were not considered to be of enough practical import to warrant making changes with urgency. In contrast, the on-going TLS 1.3 standardisation process has been highly proactive. The availability of more mature analysis tools, the threat of practical attacks, the presence of an engaged research community, and a far more open dialogue with that community have, we contend, enabled this shift in the TLS standardisation process.

This newer process has arguably been successful; several research works have helped build confidence in the protocol's design [12, 35, 36, 42, 57, 64], and others have caught flaws in a timely fashion [18, 31]. The design itself has also been significantly influenced by the research community [61], and the amount of communication between those who implement TLS and those who analyse TLS has probably never been greater.

Despite this relative success, we deem it important to reflect on whether or not the TLS 1.3 process could have been improved, and indeed to what extent it fits into the broader realm of standardisation. To this end, we briefly consider standardisation models as employed by differing standardisation bodies and examine their differences, advantages and disadvantages through the lens of TLS. Specifically, we focus on the IETF, the International Organization for standardisation (ISO) and the US government's National Institute for Standards and Technology (NIST). We conduct the thought experiment of identifying which model best suits a protocol such as TLS.

1.1 Contributions

In this paper we detail the TLS standardisation process, commenting on the recent shift in the design methodology employed by the IETF. We examine the era of post-deployment analysis, in which the IETF reacted to protocol vulnerabilities, as well as the era of pre-deployment analysis, in which the IETF is actively trying to preempt protocol weaknesses. Our contributions may be described as follows:

Pre- versus Post-Deployment Analysis. We present an account of the TLS standardisation process, examining factors which may have contributed

to the different standardisation cycles employed for TLS 1.2 and below and TLS 1.3, respectively. We comment on the tools available for analysis, levels of academic involvement, as well as the incentives driving the agents involved in the standardisation process.

Further Improvements. We comment on how the TLS 1.3 standardisation process could have been improved and present an alternative standardisation cycle for security protocols.

Comparative Analysis. We perform a comparative analysis of standardisation models and discuss the merits and faults of these models by examining their suitability for the standardisation of critical protocols such as TLS.

1.2 Related Work

In work on standardisation transparency, Griffin [49] presents the Kaleidoscope Conference case study which details actions by the International Telecommunication Union (ITU) to host an academic conference aimed at encouraging openness in standards development, as well as cultivating academia as an important external source of new ideas and technologies. We cover the concept of conferences and workshops as a means of enhancing academic involvement but also show that in the case of TLS 1.3, academia serves as an internal source of ideas in the standardisation process.

Gutmann *et al.* discuss the importance of setting requirements for security protocols in [50], another topic which we touch upon. But they appear to do so in the post-analysis setting, as they discuss formal techniques for updating requirements in response to flaws found in already published standards.

We are unaware of any work covering the complete TLS standardisation process.

1.3 Paper Organisation

In Section 2 we briefly present background on TLS and the IETF. In Section 3 we discuss the standardisation process for TLS 1.2 and below. We cover the process that has been followed for TLS 1.3 in Section 4. In Section 5 we consider the standardisation of protocols beyond the realm of the IETF and we conclude in Section 6.

2 Background

2.1 TLS

We provide a high-level overview of the TLS protocol, describing only what is relevant to the standardisation discussions to follow. We direct the reader to [34] and [78] for further details.

TLS is a network protocol designed to provide security services for protocols running at the application layer. The primary goal of TLS is to facilitate the establishment of a secure channel between two communicating entities, namely the client and the server. The TLS protocol is made up of a number sub-protocols, the two most important being the Handshake Protocol and the Record Protocol. The Handshake Protocol negotiates all cryptographically relevant parameters (including what TLS version, what authentication and key exchange method, and what subsequent symmetric key algorithms will be used). It authenticates one (or both) of the communicating entities, and establishes the keys for the symmetric algorithms that will be used in the Record Protocol to protect application data. For instance, if a client and a server agree on the `TLS_RSA_WITH_AES_128_CBC_SHA256` cipher suite during a TLS 1.2 handshake, then the server will provide an RSA certificate to be used for key exchange and entity authentication purposes. In this example, the Record Protocol will then make use of AES in CBC mode for the encryption of application data, and SHA-256 will be used in the HMAC algorithm to provide message authentication.

TLS 1.2 and below. The message flows for an initial TLS 1.2 handshake are depicted in Figure 1. Messages marked with an asterisk are optional or situation-dependent and braces of the type "[...]" indicate encryption with the application traffic keys. The client and the server exchange `ClientHello` and `ServerHello` messages in order to agree on a cipher suite and to exchange nonce values. The communicating entities also exchange cryptographic parameters (`ServerKeyExchange`, `ClientKeyExchange`) that allow for the derivation of the `pre-master secret`. Certificates and the corresponding verification information (`Certificate`, `CertificateVerify`) are sent for the purposes of entity authentication. A `master secret` is derived from the nonce values and the `pre-master secret`, and in turn used in the derivation of the application traffic keys to be employed by the Record Protocol. The `Finished` message comprises a MAC over the entire handshake, ensuring that the client and the server share an identical view of the handshake and that an active attacker has not altered any of the handshake messages.

The Handshake Protocol runs over the Record Protocol, initially with null encryption and MAC algorithms. The `ChangeCipherSpec` messages signal the intent to start using newly negotiated cryptographic algorithms and keys; they are not considered part of the handshake but instead are the messages of a peer protocol, the ChangeCipherSpec protocol. Because the `Finished` messages come after the `ChangeCipherSpec` messages, they are protected using the application data traffic keys derived in the handshake. These messages, then, are the first to be protected as part of the Record Protocol. They are followed by application data messages, now protected by the Record Protocol.

The cryptographic parameters established in the initial handshake constitute a TLS *session*. A session can be updated via a renegotiation handshake. This is a full handshake that runs under the protection of an already established TLS session. This mechanism allows cryptographic parameters to be changed (for

example, upgraded), or client authentication to be demanded by a server. In order to avoid the expensive public key operations in repeated handshakes, TLS also offers a lightweight resumption handshake in which a new `master secret` is derived from the old `pre-master secret` and new nonces, thus forcing fresh application data keys. Each such resumption handshake leads to a new TLS *connection* within the existing session; many connections can exist in parallel for each session.

The Record Protocol, as already indicated, provides a secure channel for transmission of Application Data (as well as Handshake Protocol and Alert messages). In TLS 1.0 and 1.1, it uses a "MAC-then-Encode-then-Encrypt" (MEE) construction, with the MAC algorithm being HMAC instantiated with a range of hash functions and the encryption algorithm being instantiated with CBC-mode of a block cipher or the RC4 stream cipher. Sequence numbers are included in the cryptographic processing, creating a stateful secure channel in which replays, deletions and re-orderings of TLS records can be detected. TLS 1.2 added supported for Authenticated Encryption with Associated Data (AEAD) schemes, with AES-GCM being an increasingly popular option.

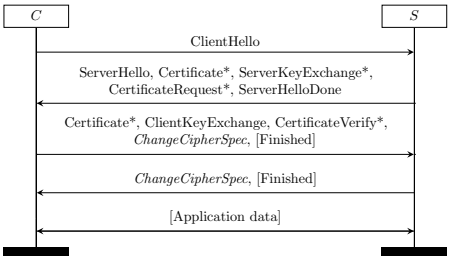

Fig. 1. TLS 1.2 handshake

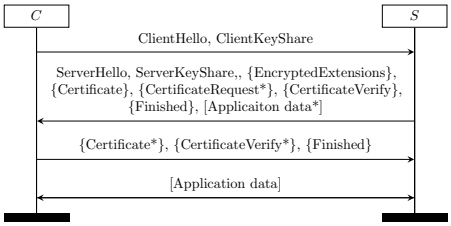

Fig. 2. TLS 1.3 (EC)DHE handshake

TLS 1.3. We provide a brief description of TLS 1.3 as defined in the current draft 15 of the standard [78], deferring discussion of the design rationale to Section 4. The design process for TLS 1.3 is on-going and several more drafts can be expected before it is complete. However, at the time of writing, the major components of the protocol appear to be fairly stable.

The message flows for an initial TLS 1.3 ephemeral Diffie-Hellman handshake are depicted in Figure 2. Messages marked with an asterisk are optional or situation-dependent. Braces of the type "{...}" indicate protection under the handshake traffic key and braces of the type "[...]" indicate protection under the application traffic key. The client and the server exchange `ClientHello` and `ServerHello` messages in order to agree a cipher suite and to exchange nonce val-

ues. The entities also exchange freshly generated Diffie-Hellman (DH) key shares along with the associated set of groups (`ClientKeyShare`, `ServerKeyShare`).

The server's first message flight will also contain extensions not used for key establishment (`EncryptedExtensions`) as well as optional early application data. Certificates and the corresponding verification information (`Certificate`, `CertificateVerify`) are exchanged for the purposes of entity authentication; the client will provide this information if requested to do so by the server (`CertificateRequest`). The `Finished` messages comprise MACs over the entire handshake transcripts using a handshake traffic key derived from the DH key shares. These messages provide integrity of the handshake as well as key confirmation. As depicted in Figure 2, the `Finished` messages are encrypted with handshake traffic keys, and no longer with application traffic keys as was the case in TLS 1.2 and below. The first records to be protected by the Record Layer are application data messages. TLS 1.3 only allows the use of AEAD schemes for the protection of this data.

Entities may also choose to use a pre-shared key (PSK) (`PreSharedKey`), or may make use of a PSK/DH combination for key exchange. In TLS 1.3, session resumption makes use of PSKs; the same is true for transmission of early client data, with the PSK used in both cases being established in an earlier handshake. This so-called *zero round-trip time (0-RTT)* capability allows the client to transmit data as part of its first flight of messages. Details pertaining to these handshake modes can be found in [78]. The renegotiation handshake as described in the TLS 1.2 RFC is no longer available in TLS 1.3.

2.2 The IETF

The IETF is a self-organized group of software developers, implementers, vendors and researchers focused on creating and maintaining engineering standards for the Internet. The IETF's mission is, simply, "to make the Internet work better" [2]. Participation by individuals is entirely voluntary and there is no formal membership or associated membership fees. The standardisation work done by the IETF is organised into areas, each of which contains several Working Groups (WGs). These areas cover all protocol layers, starting from IP [75] at the internet layer up to general application layer protocols such as HTTP [15], making the IETF the de facto technical forum for all matters concerning Internet protocol standards. The TLS WG falls into the Security Area of the IETF.

The IETF's standards are published free of charge as *Request for Comment* documents (RFCs). These are compiled using inputs from the WG mailing lists and the face-to-face discussions held at IETF meetings throughout the year. The TLS WG mailing list is remarkably active and serves as an important platform for discussion regarding the TLS RFCs. Once published, RFCs may be augmented via the use of *extensions*. These are RFCs intended to provide increased functionality and/or, in the case of TLS, security enhancements.

The IETF follows an open model of standards development. There are no barriers to entry with regards to membership and contributions, and there is a many-to-one development philosophy: all contributions are pooled in the

production of one standard, with a consensus-based process being used to decide between competing options. Analysis of TLS results from a mixture of internal and external sources; WG members may provide analyses at formal IETF meetings or on the mailing list, and research originating outside of the WG may also be consulted. In reality, and in particular for TLS 1.3, the official IETF processes have been supplemented by a shadow process involving input from a small and time-varying group of cryptographic protocol design experts. This input has been fed in to the draft editor via e-mail and in informal meetings at various conferences and workshops. Since the IETF charges no fee for its standards there are no financial barriers to adoption.

3 Post-Deployment Analysis

The standardisation process for TLS 1.2 and below can arguably be described as reactive. Following the announcement of attacks against the protocol, the TLS WG has responded by either making the necessary changes to the next version of the standard or by releasing interim recommendations or extensions. This conforms to what we will term the *design-release-break-patch* cycle of standards development. In what follows, we outline this development process as it pertains to TLS, highlighting attacks against the protocol and the IETF's responses to these attacks. We focus on attacks against the protocol rather than attacks on specific implementations (though the importance of these, for example Heartbleed and various certificate processing vulnerabilities, should not be underestimated).

We note that each TLS version builds on the previous version, incorporating changes where necessary. All TLS versions are currently in use, with clients and servers often supporting more than one version. At the time of writing, almost 98% of sites probed in the SSL Pulse survey[1] support TLS 1.0, with support of TLS 1.1 and TLS 1.2 both being in the region of 80%.

3.1 Design, Release, Break, Patch

The TLS standard officially sprang to life with a decision by the IETF to standardize a version of the Secure Sockets Layer (SSL) protocol[2] in 1996. The growing need to support e-commerce and hence the growing deployment of the SSL protocol prompted the IETF to this course of action. At this stage, two versions of SSL existed in the public domain, namely SSLv2 and SSLv3 [43]. SSLv2 had a number of weaknesses, in particular offering no defence against downgrade attacks. It was finally deprecated by the IETF in [84], published in 2011.

In 1998, Bleichenbacher published an attack on RSA when encryption used the PKCS #1 encoding scheme [26], affecting SSLv3. The attack targets the RSA-encrypted pre-master secret sent from client to server (see Section 2.1) by

[1] https://www.trustworthyinternet.org/ssl-pulse/
[2] Designed by Netscape Communications in the 1990s.

using the distinctive server-generated PKCS #1-padding error message as an oracle. Successive, adaptive calls to this oracle allow an attacker to narrow in on the value of the pre-master secret, and once this is obtained, the attacker is able to derive the symmetric keys used in the connection. The TLS 1.0 standard [32] briefly addresses this attack in a two-paragraph note that describes the following countermeasure: a server that receives an incorrectly formatted RSA block should use a pre-generated, random 48-byte value as the pre-master secret instead, thereby eliminating the oracle. The Bleichenbacher attack has been re-enabled (in various forms) in several works [52, 55, 69], the most recent case being DROWN [13], a cross-protocol attack targeting all versions of TLS running on servers that also support SSLv2. Surprisingly, a large number of servers still support this legacy version of the protocol.[3]

Following the release of TLS 1.0 [32], the first significant attack against TLS seems to be Vaudenay's padding oracle attack [28, 86]. This attack exploits the specific CBC-mode padding format used by TLS in its MEE construction in the Record Protocol. The TLS WG initially responded to the attack by adding an attack-specific countermeasure to the attack in the TLS 1.1 specification [33]. This was intended to equalise the running time of the reverse of the MEE processing – decryption, decoding, MAC verification (DDM). This knowingly left a small timing channel, but it was not believed to be exploitable. A decade later, in 2013, AlFardan and Paterson [8], in their Lucky 13 attack, showed that in fact it was exploitable in a sophisticated timing attack. Notably, the definitive patch against this attack required roughly 500 lines of new code in the OpenSSL implementation, illustrating the difficulty of making the DDM operations constant time. Moreover, several follow-up papers [7,10,11] have shown that variants of the attack are still mountable in certain circumstances or for certain implementations. The 2014 POODLE attack [71] on SSLv3 showed that SSLv3 was also vulnerable to a related but arguably more serious padding oracle attack, in which timing information was replaced with much more easy to measure error information. Because of weaknesses in the RC4 algorithm (that we discuss below), the only other encryption option in SSLv3, and because POODLE was essentially unpatchable, this attack left no other reasonable encryption options for SSLv3.

Following the release of TLS 1.2 [34] in 2008, we see more of a "patch" process being adopted by the TLS WG. During this time, we saw an explosion of attacks against TLS. We discuss some of these next.

In 2009 Ray and Rex more or less simultaneously discovered the TLS Renegotiation attack[4]. By exploiting the lack of a cryptographic binding between an attacker's initial handshake and a subsequent renegotiation handshake between an honest client and an honest server, the attacker is able to covince the server to interpret traffic – both the attacker-injected traffic and the honest client's traffic – as coming from the honest client. The WG's response to this attack was the

[3] At the time of writing, 7% of the roughly 150k servers surveyed by SSL pulse still do.

[4] See `http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html` for a description of the attack.

announcement of a mandatory TLS extension [79] applicable to all versions of TLS. The extension proposed including the respective `Finished` messages in the client and server renegotiation `Hello` messages, thus creating a binding between the two handshakes. Unfortunately, the Triple Handshake attack of Bhargavan *et al.* [20] resurrected the Renegotiation attack by cleverly exploiting the interaction of various TLS resumption and renegotiation handshakes. The attack completely breaks client authentication.

In 2011 Duong and Rizzo announced the BEAST[5] attack [38]. The attack affects TLS 1.0 and makes use of the chained-IV vulnerability observed by Moeller [70] and Bard [14], though it has its roots in an observation of Rogaway [80] from as early as 1995. BEAST exploits the fact that in TLS 1.0, the final ciphertext block of a CBC-encrypted record becomes the IV for the next record to be encrypted. This enables an attacker with a chosen plaintext capability to recover low entropy plaintexts. The main significance of the BEAST attack is the clever use of malicious JavaScript running in a victim's browser to realise the low entropy, chosen plaintext requirement and thereby mount an HTTP session cookie recovery attack against TLS. However, it should be noted that the attack required a zero-day vulnerability in the browser in order to obtain the required fine control over chosen plaintexts. The malicious JavaScript techniques were leveraged a year later by the same authors in the CRIME[6] attack (see [82] for a useful description of the attack). Unlike BEAST, however, CRIME exploits the compression side-channel inherent to all versions of TLS, a vulnerability noted in theoretical form by Kelsey in 2002 [54]. Interestingly, whilst the BEAST and CRIME attacks can be seen as having triggered the flood of research that followed, neither came from the academic research community, but instead from the "hacker" community (which partly explains the lack of formal research papers describing the attacks). Both attacks required a strong understanding not only of the cryptographic aspects of the protocol, but also of how the protocol is deployed in the web context.

The widespread response to CRIME was to disable TLS's compression feature. However, this does not completely solve the problem of compression-based attacks because compression can also take place at the application layer and introduce similar side-channels (see the BREACH and TIME attacks). A common response to BEAST was to switch to using RC4 as the encryption method in the Record Protocol, since a stream cipher would not be susceptible to the CBC vulnerabilities. Unfortunately, the RC4 keystream has long been known to be biased [66], and in 2013, AlFardan *et al* [9] exploited newly discovered and known keystream biases to obtain cookie recovery attacks when RC4 was used as the method of protection in TLS. Garman *et al.* [45] enhanced the statistical techniques of Al Fardan *et al.* and developed password recovery attacks that were of greater practical significance than those presented in [9]. The weaknesses in RC4 were further exploited by Vanhoef and Piessens [85] and Bricout *et al.* [27]. The IETF deprecated RC4 in March 2015 in [74]. Its usage has dropped rapidly as a

[5] Browser Exploit Against SSL/TLS
[6] Compression-Ratio Info-leak Made Easy

consequence of the high profile nature of the attacks, the deprecation, and the decision by major vendors to disable RC4 in their browsers.[7]

Other notable attacks to follow the BEAST, CRIME and RC4 attacks include the FREAK [17] and Logjam [6] attacks of 2015, and the SLOTH attack [24] of 2016. Both FREAK and Logjam exploit the enduring widespread support for weak export-grade cryptographic primitives. Whereas the FREAK attack affects certain TLS implementations, the Logjam vulnerability, in contrast, is the result of a protocol flaw and targets Diffie-Hellman key exchange in TLS. The attack requires a server to support export-grade cryptography, and for the client to be willing to use low security Diffie-Hellman groups. An active attacker can convince the server to provide an export-grade 512-bit group to a client that has requested a non-export DHE cipher suite, and the client will in turn accept this weak group as being valid for the DHE handshake. Clever use of a pre-computation phase for state-of-the-art discrete logarithm algorithms in [6] allowed for the quick computation of individual connections' secrets. An early intimation of these types of cross-cipher-suite attack can be found in the work of Wagner and Schneier [87] as early as 1996. The warning from this paper seems to have been either forgotten or ignored in subsequent developments of TLS. Moreover, from version 1.1 onwards, export-grade cipher suites were not supported by the TLS standards. However, as already noted, almost all servers do support TLS 1.0 and so become vulnerable to this class of attack.

The change in TLS 1.2 from supporting the MD5/SHA-1 hash function combination to supporting single hash functions for digital signatures meant that stronger hash functions such as SHA-256 could be supported but alas, so could weaker hash functions, such as MD5. Wang and Yu [88] described collision attacks against MD5 in 2005; the SLOTH attack [24] exploits this weakness to break client authentication in TLS 1.2 when MD5-based signatures are employed. The attacks presented are near-practical and falsify the belief of some practitioners that only second-preimage resistance is required of the hash functions used for TLS signatures.

We have described, at a high-level, a number of the most prominent attacks on TLS and the TLS WG's responses to these attacks. We now turn to examining whether or not these attacks were adequately addressed, and indeed, to what extent they could have been addressed by the standardisation process.

3.2 Effective Fixing, Implementation Constraints and Time Lags

The TLS 1.2 specification provides the following cautionary note with regards to the Bleichenbacher attack:

[7] See, for example, `http://www.infoworld.com/article/2979527/security/google-mozilla-microsoft-browsers-dump-rc4-encryption.html`.

> "a TLS server MUST NOT generate an alert if processing an RSA-encrypted premaster secret message fails, or the version number is not as expected. Instead, it MUST continue the handshake with a randomly generated premaster secret. It may be useful to log the real cause of failure for troubleshooting purposes; however, care must be taken to avoid leaking the information to an attacker (through, e.g., timing, log files, or other channels.)"

Upon first glance, the countermeasure appears adequate. However, as pointed out by Jager *et al.* [52], the discovery of new side-channels and the development of more sophisticated analysis techniques allow for the implementation of Bleichenbacher-style attacks even though the vulnerability was thought to be successfully patched. The attacks by Meyer *et al.* [69] on implementations of TLS serve as an example of this. One course open to the TLS WG was to remove the use of the PKCS#1 v1.5 encoding scheme in favour of the PKCS#1 v2.1 encoding scheme (implementing OAEP padding). This would have been more secure against the Bleichenbacher attack and all envisionable variants. However, as is explained in the TLS 1.1 and TLS 1.2 RFCs, in order to maintain compatibility with earlier TLS versions, this replacement was not made. We presume that the desire to maintain backwards compatibility and confidence in the *ad hoc* countermeasure trumped the evidently better security available from the use of PKCS#1 v2.1.

A very similar situation pertains to padding oracle attacks and Lucky 13: an implementation patch was put in place in TLS 1.1 and 1.2, but shown to be inadequate by the Lucky 13 attack [8]. With hindsight, it would have been less effort overall, and less damaging to the reputation of the protocol, to reform the MEE construction used in TLS at an earlier stage, replacing it with a modern design fully supported by theoretical analysis (notwithstanding the positive results of [58], whose limitations were pointed out in [73]). A repeated pattern in the development of TLS 1.2 and below is that the TLS community (a larger group of individuals and organisations than the TLS WG) seem to need to see concrete working attacks before addressing a potential vulnerability or adopting an intrinsically more secure solution, rather than applying a patch to each specific vulnerability.

In the case of attacks that exploit the existence of primitives or mechanisms that have long been known to exhibit weaknesses, the simple (but naive) solution is to simply consider removing a primitive or mechanism as soon as it is shown to be weak. However, this might not be straightforward given implementation and interoperability constraints. In the case of FREAK and Logjam, the standardisation process cannot be faulted: the weak export cipher suites were removed from TLS 1.1 and TLS 1.2 and these attacks exist as a result of poor implementation choices by practitioners. Similar remarks apply to the IV-chaining vulnerability, which while already known in 1995, was introduced to TLS 1.0 in 1999, but then removed in TLS 1.1 in 2006. Unfortunately, deployed versions of TLS did not move so quickly, with widespread support for TLS 1.0 in servers even today. On the other hand, all modern browsers will now prefer TLS 1.2 and AEAD cipher

suites in an initial handshake attempt, thanks to the long line of attacks on TLS's CBC-mode and RC4 options. In the case of SLOTH, however, the issue might not be as clear-cut. MD5-based signature schemes should not have been re-introduced in the TLS 1.2 RFC. And RC4 has a very long track-record of weaknesses stretching back more than 15 years, meaning that its phasing out from TLS could arguably have been initiated much sooner than it was, instead of waiting for the attacks to become so powerful. In many cases, particularly where hardware support for AES is available, AES-GCM could have served as a better choice for encryption.

With the many research papers professing the security of the TLS Handshake Protocol, the existence of attacks exploiting the interaction of various TLS handshakes may have come as a surprise to the TLS community. However, even here, there were early signs that things were amiss with the 1996 cross-cipher-suite attack of Wagner and Schneier [87]. Perhaps the lack of a practical attack in that paper and in later papers such as [68] led to a more relaxed attitude being adopted by the TLS WG here. The subtle interactions of different TLS handshakes was never fully considered in any analysis of TLS prior to the Triple Handshake attack of 2014. It is therefore not surprising that attacks of this form would have slipped through the standardisation process. Yet it should be remembered that the Triple Handshake attack is a resurrection of the Renegotiation attack from 2009. This is indicative of insufficiently broad or powerful analysis having been available to the TLS WG in the period intervening between the two attacks.

We argue that, in general and in view of the extreme importance of TLS, a much more conservative approach to dealing with attacks on TLS is warranted. We do, however, appreciate that bringing about meaningful change is challenging given the large scale and wide diversity of TLS deployment, the historical reticence of the major implementations to code newer versions of the protocol (especially TLS 1.2), and the slowness with which users (particularly on the server side) have tended to update their TLS versions.

3.3 Impact and Incentives

In the design-release-break-patch standardisation cycle, maximal reward for researchers has come in the form of producing and promoting high impact attacks against TLS, and engagement of the research community was largely encouraged in a retroactive fashion. The obvious problem with this incentive model is that it leaves users of published standards vulnerable to attack and imposes a potential patch action on the TLS WG. In the next section we consider whether or not a shift in the standardisation cycle leaves the opportunity for researchers to have impact (of a different kind) whilst positively benefiting the standardisation process.

4 Pre-Deployment Analysis

In contrast to the development of TLS 1.2 and below, the standardisation process for TLS 1.3 has been proactive in nature. It has followed what we describe

as the *design-break-fix-release* cycle for standards development. Working more closely with the research community, the TLS WG has released multiple protocol drafts and welcomed analyses of the protocol before its final release. As the next section will show, this design philosophy has simultaneously led to the discovery of weaknesses and provided confidence in the WG's design decisions. We explore the factors that have enabled this newer process by considering the improvements in the protocol analysis tools available, as well as the shift in design attitudes and incentives. This approach, however, has not been without its complications. In what follows we also address the challenges inherent to such an approach and comment on ways in which the process, a far as TLS is concerned, could have been improved.

4.1 Design, Break, Fix, Release

The two broad design goals for TLS 1.3 are (i) to improve efficiency of the Handshake Protocol and (ii) to address the weaknesses identified in TLS 1.2 and below.[8] The initial challenge for the TLS WG was to go about achieving these goals without having to invent an entirely new protocol: in addition to requiring new code libraries, a new protocol might introduce new weaknesses. The development of Google's QUIC Crypto by Langley and Chang [63] in 2013, offering a zero round-trip time (0-RTT) capability for the QUIC protocol [81], put pressure on the TLS WG to consider ways of reducing handshake latency in TLS 1.3. And, after the flurry of attacks in the preceding years, the protocol was due an overhaul to remove weak or broken features.

In comparison to TLS 1.2 and below, the first few drafts of TLS 1.3 (beginning with draft 00 in April 2014) incorporated changes that aim to fortify the protocol against known attacks, such as the removal of support for compression, as well as the removal of static RSA and Diffie-Hellman key exchange mechanisms, leaving ephemeral Diffie-Hellman as the only method of key exchange. Handshake latency was also reduced by the introduction of a one round-trip time (1-RTT) TLS handshake (previously an initial handshake required two round trips before a client and a server could start exchanging application data).

Two important changes that were introduced in the drafts up to and including `draft-05` are the concept of a *session hash* and the removal of the renegotiation handshake. At the time of release of `draft-04`, the session hash constituted a hash value of all messages in a handshake starting with the `ClientHello`, up to and including the `ClientKeyExchange`. The session hash is then included in the key derivation process to prevent an active attacker from synchronizing the `master secret` across two different sessions, a trick employed in the Triple Handshake attack [21]. The removal of renegotiation prevents renegotiation-based attacks, the Triple Handshake attack again serving as an example of this class of attack.

[8] See the TLS WG charter at `https://datatracker.ietf.org/wg/tls/charter/` for further details.

In terms of analysis of TLS 1.3, Dowling *et al.* [35] and Kohlweiss *et al.* [57] published works on `draft-05`, the latter set of authors using a constructive-cryptography approach to provide security guarantees for the protocol. Their work highlights that the design choice in TLS 1.3 to separate out the Handshake and Record protocols helps with their analysis, and indeed with provable security approaches in general. (Recall that in TLS 1.2 and below, the application traffic keys derived in the Handshake Protocol were used to encrypt the `Finished` messages of the Handshake Protocol itself. This interaction adds significant complexity to analyses of TLS 1.2 and below, in particular because it violates the standard indistinguishability security goal for a key exchange protocol.)

Dowling *et al.* [35] used the multi-stage key exchange model of Fischlin and Günther [41] to show that the keys output by the Handshake Protocol could be securely used in the Record Protocol. Their work provided several comments on the design of TLS 1.3, thereby explicitly providing useful feedback to the TLS WG.

In `draft-07` we see the most radical shift away from TLS 1.2, with the cryptographic core of the TLS handshake becoming strongly influenced by the OPTLS protocol of Krawczyk and Wee [62], with many OPTLS elements being incorporated into the draft. OPTLS has been expressly designed to be simple and modular, offering a 1-RTT, forward secure TLS handshake that employs ephemeral Diffie-Hellman key exchange. OPTLS also offers 0-RTT support as well as a pre-shared key (PSK) mode, capturing the use case in which a client and a server enter into the protocol having previously shared a key. This particular mode is of relevance from `draft-07` onwards as the TLS 1.2-style resumption mechanism is replaced with a mechanism that makes use of PSKs. This draft included a 0-RTT handshake and key derivation schedule that is similar to that of OPTLS, employing the HKDF primitive designed by Krawczyk [59]. The OPTLS designers provided a detailed analysis of their protocol in [62], again providing the TLS WG with confidence in its design choices.

However, it should be noted that significant changes were made in adapting OPTLS to meet the needs of TLS. For example, OPTLS originally assumed that servers' long term keys would be Diffie-Hellman values, in turn supported by certificates. However, such certificates are not widely used in practice today, potentially hindering deployment of TLS 1.3. Thus, in the "translation" of OPTLS into TLS 1.3, a two-level process was assumed, with the server using a traditional signing key to authenticate its long-term Diffie-Hellman value. But this created yet another real-world security issue: if an attacker can gain access to a server's signing capability just once, then he would be able to forge a credential enabling him to impersonate a server on a long-term basis. Thus it was decided to change the signature scope to also include client-supplied, session-specific information, limiting the value of any temporary access to the signing capability. This reduces the efficiency of the protocol, since now a fresh signature must be produced by the server in each handshake.

Notable changes in `draft-08` and `draft-09` of the protocol include the removal of support for MD5-based signatures as well as the deprecation of SHA-1-

based signatures, partly in response to the SLOTH vulnerability [24] and as result of pressure from practitioners and researchers to remove these weak primitives, as evidenced on the TLS mailing list [46, 47].

Cremers *et al.* [31] performed an automated analysis of TLS 1.3 using the Tamarin prover [83]. Their model covers `draft-10` and their analysis showed that this draft meets the goals of authenticated key exchange. They showed this in a symbolic model in which secrecy properties are more coarse-grained than would be the case in a computational model, but where the interaction of the different handshake components is easier to analyze. Cremers *et al.* anticipated the inclusion of the delayed client authentication mechanism in the TLS 1.3 series of drafts. This feature enables a server to request authentication at any point after the handshake has completed, reminiscent of the functionality provided by the renegotiation handshake in TLS 1.2 and below. They discovered a potential interaction attack which would break client authentication. The attack highlighted the strict necessity of expanding the session hash scope to include `Finished` messages. This prevents the attacker from replaying a client signature across sessions by binding the signature to the session for which it is intended. Their attack was communicated to the TLS WG, and `draft-11`, which officially incorporated the delayed client authentication mechanism, included the necessary fix as part of the design. In concurrent work, Li *et al.* [64] analysed the interaction of the various TLS 1.3 handshake modes in the computational setting using their "multi-level&stage" security model. They found `draft-10` to be secure in this model. The delayed authentication threat was not identified in this work preumably because this mechanism was not officially part of `draft-10`.

In February of 2016, just prior to the release of `draft-12`, the Internet Society hosted a "TLS Ready or Not?" (TRON) workshop. The workshop showcased analyses of TLS 1.3, both published and under development, bringing together members of the TLS Working Group, researchers and industry professionals with the aim of testing the readiness of TLS 1.3 in its then current form. Besides the aforementioned work by Kohlwiess *et al.*, Krawczyk and Wee, and Cremers *et al.*, there were several other presentations highlighting progress in the protocol's development, as well as the challenges still facing the TLS WG. Dowling *et al.* updated their previous analysis to cover `draft-10` [36], showing the full (EC)DHE handshake to be secure in the multi-stage key exchange setting. Bhargavan *et al.* introduced ProScript [18], a JavaScript variant of their verified TLS implementation, miTLS [3, 22]. Interestingly, ProScript also allows for the extraction of a symbolic model for use within the ProVerif protocol analysis tool [4, 25]. This work highlighted the potential dangers of incorporating certificate-based authentication into PSK handshakes, a potential protocol extension being considered by the TLS WG. Work on the secure of implementation of TLS 1.3 by Berdouche *et al.* [16] considered how to maintain compatibility with current TLS versions whilst protecting against downgrade attacks, and highlighted simplifications to the protocol which could be beneficial from an implementation point of of view.

Importantly, the TRON workshop led to discussions between the WG and the research community regarding potential simplifications and enhancements to the

protocol. Some of these discussions are still ongoing and have informed subsequent drafts of the protocol. The workshop also fostered an in-depth discussion regarding the security requirements for TLS 1.3. This has led to a call for contributions from researchers and practitioners alike [5]. It may seem surprising that security requirements analysis was taking place at such a late stage in the process. We comment on this further below.

At around the same time as the TRON workshop, an analysis by the Cryptographic protocol Evaluation towards Long-Lived Outstanding Security (CELLOS) Consortium, using the ProVerif tool, was announced on the TLS WG mailing list [12, 67]. This work showed the initial (EC)DHE handshake of `draft-11` to be secure in the symbolic setting.

Further publications of relevance to TLS 1.3 include the work on downgrade resilience by Bhargavan *et al.* [19] and the work on key confirmation by Fischlin *et al.* [42]. The first provides suggestions on how to strengthen downgrade security in TLS 1.3 and the second provides assurances regarding the key confirmation mechanisms used.

A smaller *ad hoc* meeting informally called "TRON2" took place in May 2016. At this meeting, the latest changes to the protocol were discussed, further formal analysis was presented, and TLS 1.3 implementations were compared.[9]

4.2 Available Tools

Since the release of TLS 1.2 in 2008, cryptographic protocol analysis tools have developed and matured to the extent that they can now effectively serve a proactive standardisation process, thereby contributing to, and perhaps even enabling, a more collaborative design effort for TLS 1.3. Significant advances have been made across all fronts, from lower-level primitives such as key derivation and authenticated encryption, to higher level primitives such as authenticated key exchange and cryptographic modelling of secure channels.

An early analysis of the TLS protocol itself can be found in the work of Gajek *et al.* [44] in 2008. However, their analysis only covers unauthenticated key exchange. Many refinements and advances in the area of provable security for TLS have since been made. A major on-going challenge has been to provide accurate modelling of the protocol and to capture the complexity of its many interacting components and modes. In 2010, Morrissey *et al.* [72] also analysed the TLS Handshake Protocol. However, their work only considered a truncated version of the protocol (with no encryption of `Finished` messages), assumed that a CCA-secure encryption scheme was used for key transport (which is unrealistic given that TLS implementations employ PKCS#1 v1.5-based RSA encryption), and relied on the random oracle model. In 2012, Jager *et al.* [51] introduced the Authenticated and Confidential Channel Establishment (ACCE) security model in an attempt to handle the unfortunate mixing of key usage in the Handshake and Record protocols; they used the ACCE model to analyse certain Diffie-Hellman-based key exchanges in TLS. Their work built in part on

[9] See `https://www.mitls.org/tron2/` for details.

a 2011 work of Paterson *et al.* [73], who introduced the notion of length-hiding Authenticated Encryption, which models desired security goals of the TLS Record Protocol. Further important works include those by Krawczyk *et al.* [60] and Kohlar *et al.* [56]. The former work analysed multiple, different TLS key exchange methods using a single, uniform set of proof techniques in the ACCE setting, while the latter extended the work of Jager *et al.* to show that the RSA and DH handshakes can be proven secure in the mutual authentication setting. Li *et al.* [65] performed a similar task for pre-shared key cipher suites. Giesen *et al.* [48] explicitly consider multiple Handshake protocol runs and their interactions in their formal treatment of the security of TLS renegotiation, while Dowling and Stebila [37] examined cipher suite and version negotiation in TLS. All of these works offer techniques that could be harnessed, and potentially extended, in the analysis of TLS 1.3, prior to its final release. Moreover, they represent a growth in interest in the TLS protocol from the research community, a necessary precursor to their greater involved in the TLS 1.3 design process.

A major step forward in the domain of program verification for TLS came with the first release of the miTLS reference implementation in 2013 [3, 22]. The miTLS implementation integrates software security and computational cryptographic security approaches so as to obtain security proofs for running code. This approach aims to eliminate the reliance on the simplifying assumptions employed by the more traditional provable security techniques – those tend to analyse abstract and somewhat high-level models of TLS and tend to ignore many implementation details in order to obtain tractable models (in the form of pseudo-code) suitable for the production of hand-generated proofs; moreover, they tend to focus on "fragments" of the TLS protocol suite rather than the entire system. Using this approach, Bhargavan *et al.* provided an epoch-based security analysis of the TLS 1.2 handshake as implemented in miTLS [23]. The miTLS implementation provides a reference for the secure implementation of TLS 1.2 and below, and interoperates with all major web browsers and servers. Not only has the miTLS project lead to the discovery of vulnerabilities such as the Triple Handshake attack and FREAK, but it has also left the TLS community with tools such as FlexTLS [1] which allows for the rapid prototyping and testing of TLS implementations. These tools are now being harnessed to assess TLS 1.3.

The rise of automated protocol analysis tools such as ProVerif [4] and the Tamarin Prover [83] can also be counted as a boon for the TLS WG. The more recent Tamarin tool, for instance, offers exceptional support for DH-based protocols and allows for the instantiation of an unbounded number of protocol participants and sessions, making it a good choice for the modelling and consequent symbolic analysis of TLS 1.3. Once established, this type of model can also be easily adapted in response to protocol changes, making this tool invaluable in an ongoing development process.

The advances in the areas of provable security, program verification and formal methods have contributed to a development environment in which a design-break-fix-release stadardization cycle can thrive. Previously, the absence of these techniques, or the limited experience in applying them to real protocols

like TLS, would have limited the amount of pre-release analysis that could have been performed, making a design-release-break-patch standardisation cycle understandable, natural even, for TLS 1.2 and below.

4.3 Involvement, Impact and Incentives

In the development of TLS 1.3, the WG has taken many positive steps in aiming to protect the protocol against the various classes of attacks mentioned in Section 3. Removal of support for weak hash functions, renegotiation, and non-AEAD encryption modes, as well as the introduction of the session hash mechanism serve as illustrative examples. The WG has also made design choices that have eased the analysis of the protocol, such as making a clean separation of the Handshake and Record Protocols, for instance. This is undoubtedly a positive step by the WG to respond to the research community's needs, marking a shift in the WG's design mindset. The TRON workshop also displays a desire by the WG to involve the research community in the design of TLS 1.3, and to incorporate its contributions. The research community, on the other hand, has gained a much greater awareness of the complexities of the TLS protocol and its many use cases, and has tried to adapt its analyses accordingly. In view of the rising interest in and focus on TLS in the research community over a period of years, and the attendant refinement of its analysis tools, this community has been in a much better position to contribute to the TLS 1.3 design process than it was for former editions of the protocol.

The ability to adapt the protocol in response to potential attacks, such as those identified by Cremers *et al.* [31] and Bhargavan *et al.* [18], makes for a stronger protocol and has allowed the WG to implement changes pre-emptively, hopefully reducing the need to create patches post-release. In comparison to the previous process described in Section 3, the design-break-fix-release standardisation cycle appears to leave the incentives for researchers unchanged, with a number of top-tier papers being produced prior to the protocol's finalisation. However, it's notable that these papers provide positive security results about TLS 1.3 rather than new attacks. We consider this to be as a result of the research community's stronger appreciation of the importance of TLS and its greater awareness of the value in contributing to its standardisation than in former development cycles.

4.4 Areas for Improvement

Although a positive step with regards to collaboration between researchers faced with analyzing TLS and the engineers faced with implementing TLS, the analysis-prior-to-deployment design strategy is not without its difficulties. Greater numbers of contributions, be they from researchers and/or implementers, have led to conflicting design opinions, potentially creating a greater administrative overhead for the TLS WG. The increase in uncoordinated contributions has also meant that the TLS 1.3 draft specification has become a rapidly moving target. This has increased the amount of analysis work required and has rendered some analyses 'outdated' within the space of few months, potentially frustrating those

those engaged in analysis of the protocol. The varied contributions have also created tension between the researchers looking at TLS 1.3, with those focused on implementation concerns suggesting improvements to the potential detriment of those concerned with the provable security aspects of the protocol. This has been an on-going issue in the area of key derivation and key separation, for example. The time scales for analysis could also potentially be more favourable: not only do rapid changes require quick analysis, but with the WG/IETF wanting official publication of TLS 1.3 within a few months of publishing the final (or a near-final) draft, this does not leave much time for detailed analysis of the final version of TLS 1.3. This is unfortunate for a protocol of such critical importance.

It is also the case that, due to the inevitable gaps in understanding between the scientific community and the more engineering-focussed participants in the TLS WG, there is the potential for miscommunication (in both directions). While we are not aware of specific instances where miscommunication or misunderstanding has seriously hampered the development of TLS 1.3, it is true that the formal security analyses presented to the WG by the research community do involve assumptions concerning attacker capabilities and the strength of the used cryptographic primitives. Sometimes these assumptions, while well understood in one community, may not be so obvious to another. One example of this would be the use of idealised cryptographic assumptions in some of the analyses based on formal methods; another (in the other direction) would be constraints on the TLS 1.3 handshake stemming from the use of Hardware Security Modules for storing server private keys.

A related area of potential improvement in the process is that of the identification of security and functionality requirements for TLS 1.3. We noted previously that only after the TRON workshop in February 2016 did it become apparent that a complete and explicit set of requirements was missing. This suggests that a different design process for TLS 1.3 could have been adopted: *requirements analysis, design, prove, release*. Instead, it appears that while some of the requirements were established early on, many others emerged only through discussion during the design phases. It is perhaps naive to hope that such a linear process would be possible for a protocol as complex as TLS, with its many use cases and with many stakeholders being involved in the development process. Certainly, multiple cycles of the "design" and "prove" steps might be needed. On the other hand, perhaps a TRON-like workshop could have been held at the commencement of the process, with the objective of flushing out the design requirements.

Finally, an issue throughout the process has been uncertainty over the degree of change permitted in TLS 1.3 relative to TLS 1.2. Initially, changes were to be incremental, potentially limiting the thinking of some participants to consider only less radical designs. Now, it is hard to argue that TLS 1.3 is anything other than a complete protocol redesign — a TLS 2.0 rather than a TLS 1.3, let us say. What novel ideas might have been brought forward had that been clear from the start?

5 Beyond TLS 1.3

Given its importance and pervasive nature, it is possible that the successful rolling out of the highly collaborative, fast-paced, proactive standardisation process may be unique to TLS. To what extent can the TLS case serve as a trendsetter, paving the way for the IETF, and indeed other standardisation bodies, to foster stronger ties with security researchers? Or is TLS an outlier in this regard? Involvement of researchers is by no means unheard of in the standardisation of security mechanisms and protocols but does the importance of TLS increase the willingness of the research community to get involved in the process?

We now examine how the newer, proactive standardisation process for TLS compares to the processes inherent to other standardisation models, such as those employed by ISO and NIST, and question to what extent these differing models would have been suitable for the standardisation of TLS. We also comment on the extent to which these models encourage active participation from the security community.

ISO. This standards body conforms to a closed model for standardisation. As with the IETF, standardisation work is organised into areas which are managed by technical committees. These committees are further broken down into subcommittees, with subcommittee 27 (SC27) being responsible for the creation and maintenance of standards concerning security techniques[10]. Within this subcommittee, WG2 is responsible for the standardisation of cryptographic mechanisms. The members of an ISO WG are not individuals but rather National Bodies (NBs) and standardisation decisions are made by the WG based on comments and contributions received from participating NBs. The formation and make-up of these NBs undoubtedly varies from nation to nation, but by and large, this type of model is characterised by barriers to entry as far as contributions are concerned as the process is far more "members only" in comparison to the open model employed by the IETF.

The development philosophy is arguably many-to-one as many members provide inputs to one standard but an ISO security standard will generally contain a number of mechanisms aimed at providing a security service, and will not be dedicated to one protocol, as is the case with the TLS RFCs. Inclusion of mechanisms in SC27 WG2 standards is usually subject to the mechanisms meeting certain maturity conditions – research from external sources is consulted and where required, NBs may perform additional analyses. This maturity requirement would potentially not suit a dynamically shifting protocol such as TLS 1.3, and the closed nature of the standardisation process potentially discourages high levels of external academic involvement. Also, ISO imposes a fee for its standards, creating a financial barrier to adoption, a less than ideal situation for a critical protocol such as TLS. The NB structure of ISO also brings into question the

[10] Other ISO subcommittees also standardise security mechanisms, such as SC17 which focuses on cards and personal identification but we focus our discussion here on SC27.

possible motives of state actors that may be involved in the standardisation process, a potential concern for a ubiquitous protocol such as TLS.

NIST. We focus here on the competition model used by NIST. This model was employed successfully in the development of AES [40] and SHA-3 [39]. This model exhibits no barriers to entry as the competitions are public and the development philosophy is one-to-one, since only one proposed candidate is selected for standardisation and contributions from the respective competitors are not pooled in the creation of the final standard. A necessity of the competition model is that algorithm/protocol requirements are clearly established and communicated. The announcement of the SHA-3 competition in the Federal Register in 2007 [77], for instance, contained sections covering minimum criteria as well as evaluation criteria. Analysis of the SHA-3 candidates was performed by NIST and the larger cryptographic community, with many comments being communicated on the public hash forum set up for the competition. Many of the analyses culminated in top-tier publications (see [29] for a comprehensive list), thereby productively serving academic incentives as part of the standardisation process. NIST also held several SHA-3 conferences as a means of obtaining public feedback.

Some of the elements of the SHA-3 standardisation narrative overlap with the TLS 1.3 standardisation process discussed in Section 4. The analysis-prior-to-deployment development methodology, the use of public mailing lists and the hosting of public conferences/workshops are all aspects in which the TLS 1.3 process is similar. But the process differs in that the requirements for TLS 1.3 were not fully expressed before the design commenced. There is of course no *explicit* element of competition, differentiating the competition model from the open model. On the other hand, individual researchers and research teams do stand to gain greatly by having their ideas adopted in TLS 1.3, whether through personal kudos or recognition that is internal (promotion, company awards) or external (prizes, paper citations). Finally, the SHA-3 competition ran for several years (from 2007 to 2012), allowing more time for detailed analysis.

Like the open model, there is no cost associated with the final product and this model could most certainly work for TLS. However, it is doubtful whether such a model would allow for the rapid development of the protocol, as we have seen with TLS 1.3. The competition model has proven to be suitable for cryptographic primitives like block ciphers and hash functions. A complex protocol such as TLS might be too large in scope for any one research team to design in its entirety, perhaps making a collaborative standardisation model more appropriate.

6 Conclusion

We have presented an account of TLS standardisation, starting with the early versions of TLS, right up until TLS 1.3, which is, at the time of writing, nearing completion. We have described how the process for TLS 1.2 and below fits the design-release-break-patch cycle of standards development and how a shift in the process has resulted in the standardisation of TLS 1.3 conforming to the

design-break-fix-release development cycle. We have commented on the factors that have influenced the shift in the TLS WG's design methodology, namely, the protocol analysis tools available, the levels of involvement from the research community, and the incentives driving the relevant stakeholders. This newer process exhibits benefits over the cycle employed previously as it allows for the preemptive detection and fixing of weaknesses, thus producing a potentially stronger protocol and reducing the need for patches post-release. We have gone on to suggest that the process for TLS 1.3 could have been enhanced even further by the WG considering a requirements analysis-design-prove-release cycle for development of the standard. We have also examined the standardisation of TLS in relation to a number of varying standardisation models. We find that the current, collaborative process under the open model of the IETF shows promise in producing a strong protocol but that the competition model as employed by NIST would also potentially have suited a protocol such as TLS.

We believe our work to be the first attempt at a TLS standardisation diegesis, and that a detailed classification of standardisation models, based on further case studies, would make for interesting future work.

Acknowledgements

Paterson was supported in part by a research programme funded by Huawei Technologies and delivered through the Institute for Cyber Security Innovation at Royal Holloway, University of London, and in part by EPSRC grant EP/M013472/1. Van der Merwe was supported by the EPSRC as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London. We thank Eric Rescorla and the anonymous reviewers of SSR 2016 for their valuable feedback on the paper.

References

1. FlexTLS: A Tool for Testing TLS Implementations. Available at: `https://mitls.org/pages/flextls`.
2. Getting Started in the IETF. `https://www.ietf.org/newcomers.html`. Accessed: 2016-06-08.
3. miTLS: A Verified Reference Implementation of TLS. Available at: `https://mitls.org/`.
4. ProVerif: Cryptographic protocol verifier in the formal model. Available at: `http://prosecco.gforge.inria.fr/personal/bblanche/proverif/`.
5. TLS 1.3 Security Properties. Available at: `https://github.com/tls13properties/tls13-properties`.
6. David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Ray et al. [76], pages 5–17.

7. Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on Amazon's s2n implementation of TLS. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 622–643. Springer, 2016.
8. Nadhem AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In Robin Sommer, editor, *Proceedings of the 2013 IEEE Symposium on Security and Privacy (S&P 2013)*, pages 526–540, San Diego, CA, USA, May 2013. IEEE Press.
9. Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the Security of RC4 in TLS. In Samuel T. King, editor, *Proceedings of the 22nd USENIX Security Symposium*, pages 305–320, Washington D.C., USA, August 2013. USENIX.
10. José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, and François Dupressoir. Verifiable side-channel security of cryptographic implementations: Constant-time MEE-CBC. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 163–184. Springer, 2016.
11. Gorka Irazoqui Apecechea, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Lucky 13 strikes back. In Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn, editors, *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 85–96. ACM, 2015.
12. K. Arai. Formal Verification of TLS 1.3 Full Handshake Protocol Using Proverif. Technical report, Cryptographic protocol Evaluation toward Long-Lived Outstanding Security Consortium (CELLOS), February 2016. Available at: `https://www.cellos-consortium.org/studygroup/TLS1.3-fullhandshake-draft11.pv`.
13. Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: breaking TLS using sslv2. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 689–706. USENIX Association, 2016.
14. Gregory V. Bard. A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT*, pages 99–109. INSTICC Press, 2006.
15. T. Berners-Lee, R. Fielding, and H. Frystyk. The Hypertext Transfer Protocol HTTP/1.0. RFC 1945 (Informational), May 1996.
16. B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, S. Ishtiaq, M. Kohlweiss, J. Protzenko, N. Swamy, S. Zanella-Bguelin, and J.K. Zinzindohou. Towards a Provably Secure Implementation of TLS 1.3. Presented at TRON 1.0, San Diego, CA, USA, February 21, 2016.
17. Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 535–552. IEEE Computer Society, 2015.

18. K. Bhargavan, N. Kobeissi, and B. Blanchet. ProScript TLS: Building a TLS 1.3 Implementation with a Verifiable Protocol Model. Presented at TRON 1.0, San Diego, CA, USA, February 21, 2016.
19. Karthikeyan Bhargavan, Christina Brzuska, Cèdric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella-Bèguellin. Downgrade Resilience in Key-Exchange Protocols. In *2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 23-25, 2016*, 2016.
20. Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 98–113. IEEE Computer Society, 2014.
21. Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 98–113, 2014.
22. Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing TLS with verified cryptographic security. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 445–459. IEEE Computer Society, 2013.
23. Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella Béguelin. Proving the TLS handshake secure (as it is). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 235–255. Springer, 2014.
24. Karthikeyan Bhargavan and Gaëtan Leurent. Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.
25. Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*, pages 82–96, 2001.
26. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.
27. Remi Bricout, Sean Murphy, Kenneth G. Paterson, and Thyla Van der Merwe. Analysing and exploiting the Mantin biases in RC4. *IACR Cryptology ePrint Archive*, 2016:63, 2016.
28. Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password Interception in a SSL/TLS Channel. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer, 2003.
29. S. Chauhan, R. Sobti, G. Geetha, and S. Anand. Cryptanalysis of SHA-3 Candidates: A Survey. *Research Journal of Information Technology*, 5:149–159, 2013.
30. Liqun Chen and Chris J. Mitchell, editors. *Security Standardisation Research - First International Conference, SSR 2014, London, UK, December 16-17, 2014. Proceedings*, volume 8893 of *Lecture Notes in Computer Science*. Springer, 2014.

31. Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication. In *2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 23-25, 2016*, 2016.
32. T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, Internet Engineering Task Force, January 1999.
33. T. Dierks and C. Allen. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Internet Engineering Task Force, April 2006.
34. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Internet Engineering Task Force, August 2008.
35. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Ray et al. [76], pages 1197–1210.
36. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081, 2016. `http://eprint.iacr.org/`.
37. Benjamin Dowling and Douglas Stebila. Modelling ciphersuite and version negotiation in the TLS protocol. In Ernest Foo and Douglas Stebila, editors, *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*, volume 9144 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2015.
38. Thai Duong and Juliano Rizzo. Here come the $\oplus$ Ninjas. Unpublished manuscript, 2011.
39. Morris J. Dworkin. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. FIPS 202, August 2015.
40. Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, and James F. Dray Jr. Announcing the Advanced Encryption Standard (AES). FIPS PUB 197, November 2001.
41. Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1193–1204, 2014.
42. Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key Confirmation in Key Exchange: A Formal Treatment and Implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 23-25, 2016*, 2016.
43. A. Freier and P. Karlton. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic Document), August 2011.
44. Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally composable security analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security, Second International Conference, ProvSec 2008, Shanghai, China, October 30 - November 1, 2008. Proceedings*, volume 5324 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2008.
45. Christina Garman, Kenneth G. Paterson, and Thyla Van der Merwe. Attacks only get better: Password recovery attacks against RC4 in TLS. In Jung and Holz [53], pages 113–128.
46. D. Garret. banning SHA-1 in TLS 1.3, a new attempt. TLS mailing list post, October 2015. Available at `http://www.ietf.org/mail-archive/web/tls/current/msg17956.html`.

47. D. Garret. MD5 diediedie (was Re: Deprecating TLS 1.0, 1.1 and SHA1 signature algorithms). TLS mailing list post, January 2016. Available at `http://www.ietf.org/mail-archive/web/tls/current/msg18977.html`.
48. Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of TLS renegotiation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 387–398. ACM, 2013.
49. Phillip H. Griffin. Standardization transparency - an out of body experience. In Chen and Mitchell [30], pages 57–68.
50. Joshua D. Guttman, Moses D. Liskov, and Paul D. Rowe. Security goals and evolving standards. In Chen and Mitchell [30], pages 93–110.
51. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 273–293, 2012.
52. Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1185–1196, 2015.
53. Jaeyeon Jung and Thorsten Holz, editors. *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. USENIX Association, 2015.
54. John Kelsey. Compression and information leakage of plaintext. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2002.
55. Vlastimil Klíma, Ondrej Pokorný, and Tomás Rosa. Attacking RSA-Based Sessions in SSL/TLS. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pages 426–440, 2003.
56. Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DH and TLS-RSA in the standard model. *IACR Cryptology ePrint Archive*, 2013:367, 2013.
57. Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. (De-)Constructing TLS. *IACR Cryptology ePrint Archive*, 2014:20, 2014.
58. Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.
59. Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 631–648, 2010.
60. Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 429–448, 2013.
61. Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. *IACR Cryptology ePrint Archive*, 2015:978, 2015.
62. Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 81–96. IEEE, 2016.
63. A. Langley and W. Chang. QUIC Crypto, June 2013. Available at `https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/`.

64. Xinyu Li, Jing Xu, Zhenfeng Zhang, Denugguo Feng, and Honggang Hu. Multiple Handshakes Security of TLS 1.3 Candidates. In *2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 23-25, 2016*, 2016.
65. Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. On the security of the pre-shared key ciphersuites of TLS. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 669–684. Springer, 2014.
66. Itsik Mantin and Adi Shamir. A practical attack on broadcast RC4. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer, 2001.
67. S. Matsuo. Formal Verification of TLS 1.3 Full Handshake Protocol Using ProVerif (Draft-11). TLS mailing list post, February 2016. Available at `https://www.ietf.org/mail-archive/web/tls/current/msg19339.html`.
68. Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneela. A cross-protocol attack on the TLS protocol. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS'12)*, pages 62–72, Raleigh, NC, USA, October 2012. ACM Press.
69. Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS implementations: New Bleichenbacher side channels and attacks. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 733–748. USENIX Association, 2014.
70. Bodo Moeller. Security of CBC ciphersuites in SSL/TLS: Problems and countermeasures. Unpublished manuscript, May 2004. `http://www.openssl.org/~bodo/tls-cbc.txt`.
71. Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This POODLE bites: Exploiting the SSL 3.0 fallback, September 2014.
72. Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The TLS Handshake Protocol: A modular analysis. *J. Cryptology*, 23(2):187–223, 2010.
73. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2011.
74. A. Popov. Prohibiting RC4 Cipher Suites. RFC 7465 (Proposed Standard), February 2015.
75. J. Postel. Internet Protocol. RFC 791, Internet Engineering Task Force, September 1981.
76. Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*. ACM, 2015.
77. Federal Register. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA3) Family. Federal Register, November 2007.
78. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, Draft 15. Internet draft, Internet Engineering Task Force, August 2016.
79. E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. RFC 5746 (Proposed Standard), February 2010.

80. Phil Rogaway. Problems with proposed IP cryptography. Unpublished manuscript, 1995. http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt.
81. J. Roskind. QUIC: Quick UDP Internet Connections, April 2012. Available at https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit?pref=2&pli=1.
82. Pratik Guha Sarkar and Shawn Fitzgerald. Attacks on SSL – a comprehensive study of BEAST, CRIME, TIME, BREACH, Lucky 13 and RC4 biases, August 2013.
83. Tamarin prover GitHub repository (develop branch). https://github.com/tamarin-prover/tamarin-prover, 2015.
84. S. Turner and T. Polk. Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176 (Proposed Standard), March 2011.
85. Mathy Vanhoef and Frank Piessens. All your biases belong to us: Breaking RC4 in WPA-TKIP and TLS. In Jung and Holz [53], pages 97–112.
86. Serge Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ... In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.
87. David Wagner and Bruce Schneier. Analysis of the SSL 3.0 protocol. In *USENIX Electronic Commerce*, 1996.
88. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.