Transport Layer Secured Password-Authenticated Key Exchange

By Phillip H. Griffin – ISSA Fellow, Raleigh Chapter

This article describes how to achieve mutual authentication using Transport Layer Security (TLS) without client certificates or major changes to the TLS protocol. Using a password-authenticated key exchange (PAKE) protocol following the TLS handshake can protect user credentials from phishing and man-in-the-middle attacks.

Abstract

This article describes how to achieve mutual authentication using Transport Layer Security (TLS) without client certificates or major changes to the TLS protocol. Using a password-authenticated key exchange (PAKE) protocol following the TLS handshake can protect user credentials from phishing and man-in-the-middle attacks. Unlike TLS, this PAKE approach does not rely on trustworthy certificate authorities (CAs), a fully functional public key infrastructure (PKI), adequate browser certificate revocation checking, or changes to user behavior or in their understanding of certificate validation.

The Transport Layer Security (TLS) protocol[1] has really been taking it on the chin lately, with one successful punch after the next landing a blow. The latest problem to emerge is FREAK (Factoring Attack on RSA-EXPORT), which follows a number of others that are still being dealt with by the Internet community. These others include POODLE, BEAST, Crime, Lucky 13, and perhaps the most famous, the Heartbleed bug, which gained a spot in the public eye with its own website[2] and logo. If those were not enough to shake our confidence in TLS, there are related problems to consider, such as CBC-mode cipher suite vulnerabilities and attacks on the widely deployed RC4 stream cipher.

These Internet security problems are cause for serious concern, and they have rightly received media attention and fostered a great deal of discussion among information security professionals. The prominence of these problems in the spotlight, however, tends to overshadow a more fundamental issue with TLS: its failure as an authentication mechanism to prevent phishing, pharming, and website spoofing attacks. These ills continue to plague us with an increasing number of victims, particularly in the United States where identity theft and fraud continue to rise each year.

According to a recent RSA fraud report,[3] economic losses alone had topped a billion dollars in year 2013. These authentication-related problems are not solely the blame of the TLS protocol itself. There is plenty of blame to go around, from the many users without a public key certificate needed to support mutual authentication, to software bugs, poor configurations, and implementation failures. Despite the recent publicity, the news is not all bad. TLS still does a great job of providing a secure channel for communications.

Perhaps our favorite security protocol just needs a little help from our old "friend," that authentication stalwart that we all love to hate and wish would go away, but that never seems quite to outlive its utility: the user password. Despite their many problems, passwords remain the "most common web authentication technique in use today" [1]. Typically, to authenticate on the web, a user types a "password directly into a webpage from the site to which she wishes to authenticate herself" using an HTML form [1].

1 TLS previously was known as Secure Sockets Layer (SSL), or referred to jointly as SSL/TLS.

2 http://heartbleed.com.

3 http://www.emc.com/collateral/fraud-report/online-rsa-fraud-report-012013.pdf.

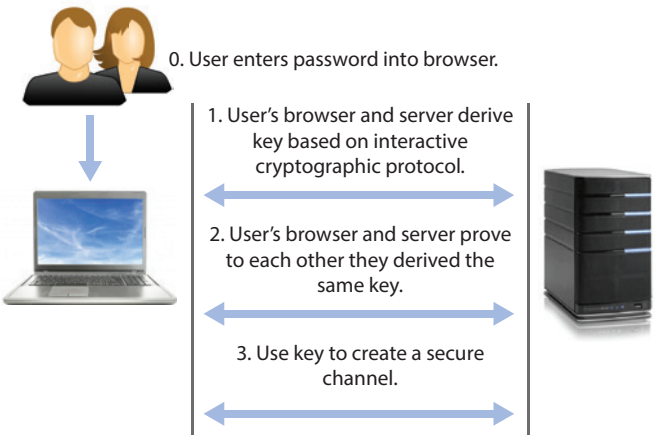

Figure 1 – PAKE-based web authentication (source – Engler, et al. [1])

Information security researchers have been looking into how to merge the best attributes of passwords and TLS, at how they might combine the convenience and ubiquity of password authentication with the secure communications channel provided by TLS. There have been failed attempts to do so in the past, and there are still several significant bumps on the road to making this marriage a reality. There are technology patents in the way, competing security standards, and differences over what might be the best implementation approach.

However, a paper presented last December at the Security Standardization Research conference[4] in London, may offer a best path forward. The approach described in "Secure Modular Password Authentication for the Web Using Channel Bindings" [2] could help fight phishing, make widespread mutual authentication a reality, and do so without changing the current TLS standard. The suggested approach applies Password Authenticated Key Exchange (PAKE) to web authentication (figure 1).

PAKE is a "cryptographic protocol that allows two parties who share knowledge of a password to mutually authenticate each other and establish a shared key, without explicitly revealing the password in the process" [1]. If the server is spoofing the user, it will not know the user password, step 2 in figure 1 will fail, and the authentication session will end without revealing the user password. The trick is in how best to combine PAKE with TLSs.

TLS protocol

TLS is a multilayer protocol that can be grouped into two distinct parts, a *handshake protocol* and a *record protocol* [3]. The record protocol follows the handshake and provides data integrity and data confidentiality services for communications over a secure channel. The handshake protocol provides entity authentication and establishes "the parameters required for subsequent communications security" [3]. Once the channel is established, users can "securely identify themselves using a short (memorable) password" [4].

A common approach used in many web applications is to "add a straightforward password check: the server sends its X.509 certificate for explicit verification, then the user sends his password through the TLS secure channel" [4]. In TLS, user software first must authenticate the server by verifying a certificate authority trusted by the user has signed the server certificate. This approach does not always end well. The user must depend on the software agent to perform certificate validation. If the server is an imposter or certificate validation is flawed, the user may reveal his password and subsequent information while all the time believing the system is protecting his secret information.

Though most secure websites rely on TLS to authenticate the server to the client, mutual authentication is an optional handshake feature less commonly used, "because not every client has a certified public key" certificate [3]. Instead, the client authenticates to the server "by sending a password to the server after the establishment of a TLS-protected channel" [3]. Web spoofing and phishing attacks rely on this fact to capture user credentials and other sensitive information [3].

Phishing effectiveness seems to be nondiscriminatory. Studies have shown that neither "education, age, sex, previous experience, nor hours of computer use showed a statistically significant correlation with vulnerability to phishing" [5]. There have been efforts to steer web surfers to safe waters, but alerts and popup warnings about fraudulent server certificates have not been effective at deterring users. Most study participants proceed "without hesitation when presented with warnings" [5].

Users tend not to notice changes in the lock icon, and they will simply click through any barriers imposed by popup warnings. Awareness training can help, but its effectiveness can be short lived. Adding PAKE as an additional control to augment TLS authentication can ensure users never reveal their passwords to imposter servers, and make mutual authentication possible without depending on users to purchase, install, and manage digital certificates.

Password-Authenticated Key Exchange

International standardization

The ISO/IEC JTC 1/ SC27 IT Security techniques[5] group published their ISO/IEC 11770-4 *Key Management – Mechanisms Based on Weak Secrets*[6] standard in 2006. The standard considers passwords weak secrets since a person can easily memorize them. ISO/IEC 11770-4 defined three password-authenticated key exchange (PAKE) mechanisms specifically designed to establish one or more secret keys based on a memorized password: balanced password-authenticated key agreement, augmented password-authenticated key agreement, and password-authenticated key retrieval.

ITU-T[7] standardized an efficient PAKE technique most similar to the "balanced" SC 27 mechanism in their X.1035[8] rec-

4 http://www.ssr2014.com/.

5 http://www.iso.org/iso/iso_technical_committee.html?commid=45306.

6 http://www.iso.org/iso/catalogue_detail.htm?csnumber=39723.

7 The Telecommunication Standardization Sector of the International Telecommunication Union (ITU).

8 http://www.itu.int/rec/T-REC-X.1035-200702-I/en.

ommendation in early 2007. Their goal was to ensure mutual authentication "in the act of establishing a symmetric cryptographic key via Diffie-Hellman exchange" [6]. Key establishment based on Diffie-Hellman ensures perfect forward secrecy, a highly desirable protocol property "that guarantees that compromise of a session key or long-term private key after a given session does not cause the compromise of any earlier session" [6]. However, achieving perfect forward secrecy requires that both parties choose fresh random values for their exponents each time they perform the protocol.

Passwords are cryptographically weak secrets that people are able to memorize. PAKE provides a strong key-exchange mechanism, even with weak passwords, but assumes that users protect their passwords and do not expose them to an attacker. Through its use of password authentication, the PAKE protocol protects users from man-in-the-middle attacks. Mutual authentication relies on a password, a previously shared secret. Due to its design, the protocol never exposes the secret password to an eavesdropper during the operation of PAKE [6]. This feature of PAKE prevents off-line dictionary attacks, a common password authentication problem.

Establishing the initial pre-shared secret (e.g., password) could occur in a number of ways, by telephone or on site (e.g., in a bank, medical facility, or in a retail store). Password establishment could occur online at a trusted enrollment webpage, relying on the public key component of a public-private key pair to protect the shared secret during transport. Both parties must know the password to establish a shared symmetric encryption key. If key establishment succeeds, both parties must have used the same password and they have achieved mutual authentication. If key establishment fails, the PAKE protocol halts without having divulged the shared secret password to a third party.

Prior to operation of the protocol, the communicating parties must agree on several values whose exchange on a network might be subject to attacks that could harm the security provided by PAKE. These pre-agreed values include the shared secret password and the publicly known Diffie-Hellman constants, p and g. The constant p should be a safe prime number large enough to make solving the discrete log problem infeasible for an attacker. The value chosen for g should ensure that "powers of g modulo p cover the entire range of p-1 integers from 1 to p-1" [6].

A few years after publication of the X.1035 standard, several ITU-T members informed the Internet community of their PAKE work by releasing an informative request for comments paper, "IETF RFC 5683 Password-Authenticated Key (PAK) Diffie-Hellman Exchange."[9] This document presented more granular security considerations for implementers, expanding on the content of the ITU-T standard, but IETF did not choose to develop PAKE as an Internet standard.

In 2012, a group of researchers based in Japan presented a standards track draft to the IETF,[10] the *Mutual Authentication Protocol for HTTP*, followed shortly by a companion standards track draft, *KAM3-based Cryptographic Algorithms*.[11] The later draft relied on one of three password-authenticated key agreement schemes defined in 2006 in the ISO/IEC 11770-4 *Key Management – Mechanisms Based on Weak Secrets*[12] standard. The draft also noted that the author's organization had filed for patents on the protocol. Neither of

9 http://www.rfc-editor.org/rfc/rfc5683.txt.

10 http://tools.ietf.org/html/draft-oiwa-http-mutualauth-12.

11 https://tools.ietf.org/html/draft-oiwa-http-mutualauth-algo-02.

12 http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39723.

these IETF drafts seems to have been progressed beyond their 2014 versions.

Adoption impediments

In their SSR 2014 conference paper, Manulis, Stebila, and Denham [2] cite patents as a primary obstruction to PAKE being "implemented in existing web browser and server technologies" [2]. This includes both "patents covering PAKE in general (some of which have recently expired in the US)" and "patents on proposed standards such as the Secure Remote Password (SRP) protocol" that could be used by implementers [2]. Though SRP has been standardized in TLS "as an alternative to public-key authenticated ciphersuites," it has not been implemented in any "major web browsers or servers" [2].

Other PAKE-based proposed solutions have had only minor success. These include PAKE by juggling (J-PAKE) and "the provably secure Simple Open Key Exchange (SOKE) cipher suite" [2]. These past proposals have shared one important attribute: they all proposed modifying the TLS standards and required changes to the TLS handshake protocol. If adopted, these approaches might have caused backward compatibility issues and negatively affected many users. Manulis, et al. [2] have proposed a different approach, one that does not require changing the TLS protocol standards. They refer to their approach as "password-authenticated and confidential channel establishment (PACCE)" [2].

PACCE protocol

The authors demonstrate that cryptographically binding "a secure-channel protocol such as TLS with a password authentication protocol" such as PAKE, using either the TLS server certificate or the transcript of the client-server TLS handshake, "results in a secure PACCE protocol" [2]. As shown in figure 2, the client initiates the PAKE protocol following successful completion of the TLS handshake, using the shared secret password and a TLS session tag to create a cryp-

The Curmudgeon

The Five Horsemen…

Thinking about what the "greatest threat" to our "critical infrastructure" might be, I come up with threats well outside the "usual suspects": Complacency, Ignorance, Obliviousness, Politics, and Budgets. These "five horsemen" ensure lifetime employment for those of us in the field.

Ignorance is merely lack of knowledge or understanding. Obliviousness is deliberate ignorance; obliviousness is chosen. Complacency is just being comfortable and not wanting to do anything new or to change. Ignorance is somewhat excusable. Obliviousness and complacency are chosen, not excusable.

Most people don't know, don't want to know, and don't want to have to think about the implications of a purchase—appliance, automobile, or much of anything else. They want convenience. They neither care nor want to learn how it works, what needs maintenance, or what it depends on. "It ought to just work."

Politics and budgets are complementary and self-reinforcing. Whenever you have budgets, there are "limits." The assumption is "there is only so much to go around." Politics are negotiations and public presentations to get "more." Whoever is "in power" gets to set the budgets, by perception restricting anyone or anything else. What do those have to do with "information security"?

People have "budgets." They want their Things inexpensive. "Things should just work." The corporation producing things chooses between "production cost" and "additional security." Both are passed through to the consumer. Changing the revenue (profit) stream would affect the rewards to the executive suite. The "competition" would undersell because "they" would ignore the additional security, contain costs, sell for less, and win market share. Security, information assurance, and testing are viewed as "cost centers." Corporations focus on profit centers.

Bringing this home to the consumers:

They get up in the morning, check the weather forecast on the display on the refrigerator, reach in and pull out breakfast. The refrigerator senses the food containers' movements and updates its shopping list. People move around in the house; the smart lights and appliances sense their movements and change their settings, reporting to each other through the automatically-set-up internetwork of "things" in the house. Everyone gathers up smart phones, watches, exercise monitors, tablets, and other personal devices and goes to work or school. The house "goes quiet," switching to "absent" mode. The smart thermostats reduce energy consumption. The smart electric meter notices the reduced load and reports to the power company.

The adults take their "smart" automobiles to work or use public transportation, showing their "charge cards" via (near field communication) physical cards, barcode cards, or maybe some "ticket" barcode on their smart phones.

The children use their "identity cards" to "log in" to the school bus. They, too, have smart phones, exercise trackers, and tablets or similar for "textbooks" at the "new" schools.

Maybe there is a baby monitor involved: a camera, various LED lights, a microphone, and some kind of link to provide the information to parents via their smart phones, tablets, or the Internet. Perhaps there is a smart lock on the house; it reacts to the presence of the smart phones or other personal devices supposedly only available to the family, but easily "joined" to the smart lock.

If you've followed security news recently, you've seen the problems with SOHO routers, access points, and firewalls becoming DDoS and spam bots. You've read about smart lights falling to the usual attacks; smart cars having been cracked; and statements about accessing flight control systems via the in-cabin entertainment systems.

They could save a lot of weight and cost using a single wiring buss to carry the traffic for both systems. Surely no one would deliberately interfere with flight control systems in flight. Besides, who would attack the entertainment system? (Who in their right mind would deliberately put safety-of-life systems on the same physical, let alone logical, buss as publicly accessible networks?!)

Budget, Politics, Ignorance, Complacency, and Obliviousness. Giddyup! Hyah!

Your local, grumpy, tie-wearing, un-impressed, and suspicious Curmudgeon

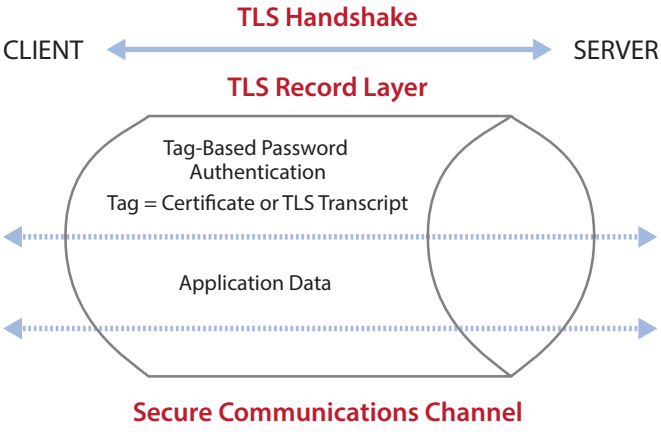

Figure 2 – High-level view of PACCE (source – Manulis, et al. [2])

tographic key used to protect a message sent to the server. If the recipient of the message encrypted by the client does not know the password, the recipient cannot create the key needed to decrypt the client message and the session ends without revealing any private client information.

The approach of the authors, as shown in figure 2, can rely on the confidentiality and data integrity services of a secure channel created using TLS, but it does not rely on TLS for server authentication. Mutual authentication is achieved if the server can decrypt the client message and encrypt a message to the client to provide confirmation that they know the shared secret password. The authors eliminate some of the common security issues encountered when relying on TLS handshake authentication, such as "problems with the trustworthiness of certification authorities (CAs), inadequate deployment of certificate revocation checking, ongoing threats from phishing attacks, and the poor ability of the users to understand and validate certificates" [2].

PAKE protocols allow a client and a server to determine whether they both know a particular password "while cryptographically hiding any information" about their shared secret. Communicating parties can resist dictionary attacks by an adversary that may have observed or participated in the PAKE protocol, since the adversary cannot test many values against the encrypted text. Once the protocol completes successfully, PAKE provides both authenticated parties with secure session keys that can be used for encryption [2].

Manulis, et al. implement a desktop version of their TLS-based prototype as an easily installed Firefox browser extension.[13] Links to a reprint of the authors SSR 2014 conference paper, the server-side code needed to test drive their work, and a demonstration program are freely available for download.[14] It would be possible to create similar implementations with secure channel protocols other than TLS, or by using other PAKE variations, such as the Simple Password Exponential Key Exchange (SPEKE) protocol, "a popular password-authenticated key exchange (PAKE) protocol that has been widely used since the middle 1990s" [7].

Conclusions

Coupling a PAKE protocol with TLS gives added protection to user passwords. PAKE protects passwords from spoofing servers using a Diffie-Hellman key agreement scheme. Servers only receive user-encrypted passwords. Servers that do

Typical TLS Secure Session

1. User (client) clicks on an HTTPS link to access a server resource such as a bank account or medical record.
2. Server performs a TLS handshake protocol with the client to authenticate the server to the client and establish a session key.
3. Server sends an HTML form to the client; user enters the account name and password previously established. The account name and password are sent to the server protected by the secure channel provided under the established session key.
4. Server authenticates the user. [If the server is an imposter or if there is a man-in-the-middle attacker, the recipient steals the user password.]

Secure TLS Session with PAKE

1. User (client) clicks on an HTTPS link to access a server resource such as a bank account or medical record.
2. Server performs a TLS handshake protocol with the client to authenticate the server to the client and establish a session key.
3. The user enters password into a locally controlled HTML form, then uses the password and a tag that identifies the TLS handshake session to create a PAKE key using a Diffie-Hellman protocol.
4. The client encrypts the password using the PAKE key; an account name and the encrypted password are sent to the server through the TLS channel; and the server attempts to authenticate the user.
5. If the recipient is not an imposter or a man-in-the-middle attacker, the server uses the password previously established with the user account to derive the same PAKE key used by the client, decrypts the client password, and authenticates the client. (Otherwise, the user password has never been revealed to an attacker, and the encrypted message recipient cannot complete the PAKE protocol by responding to the client to confirm knowledge of the user password using the derived PAKE key—the protocol ends.)
6. The server encrypts a key confirmation message using the shared PAKE key, and sends this encrypted message to the client.
7. The client decrypts the key-confirmation message and authenticates the server (and mutual authentication is achieved).
8. The client and server can now communicate using the established TLS session key with confidence in the identity of the communicating parties.

13 http://www.douglas.stebila.ca/files/research/papers/MSD14-pow-20130806.xpi.
14 http://www.douglas.stebila.ca/research/papers/msd14/.

not possess the password are unable to derive the symmetric key needed to decrypt the cipher text and prove they are not imposters.

Augmenting TLS authentication with PAKE can transform the dream of mutual authentication into a reality. Users are not required to purchase and maintain personal certificates or to understand anything about the complexities of certificate signature verification, trusted path validation, or certificate validity periods and expiration dates. Users need only remember a weak secret, their personally chosen password, to gain the benefits of mutual authentication.

Using a PAKE protocol, users can identify themselves to a server and gain assurance that the server they are trying to connect to is not an imposter. Coupling PAKE with TLS does not require changes to the TLS protocol standards. The addition of PAKE protection does not require changes in user behavior or to the user authentication experience, but can help defeat phishing and provide users with safer surfing on the web.

References

[1] Engler, J., Karlof, C., Shi, E., Song, D. Is it too late for PAKE? In *Web 2.0 Security and Privacy* (W2SP) 2009 (2009). Retrieved May 31, 2015, from http://www.ieee-security.org/TC/W2SP/2009/papers/s4p1.pdf.

[2] Manulis, M., Stebila, D., & Denham, N. (2014). Secure Modular Password Authentication for the Web Using Channel Bindings. In *Security Standardisation Research: First International Conference, SSR 2014, London, UK, December 16-17, 2014. Proceedings (Vol. 8893, pp. 167-189)*. Chen, L., & Mitchell, C. (Eds.). Springer International Publishing. Retrieved May 31, 2015, from http://www.springer.com/us/book/9783319140537.

[3] Alsaid, A., & Mitchell, C. J. (2006, July). Preventing Phishing Attacks Using Trusted Computing Technology. In *Proceedings of the 6th International Network Conference (INC'06) (pp. 221-228)*. Retrieved May 31, 2015, from http://www.cs.auckland.ac.nz/research/groups/ssg/pastbib/pastpapers/alsaid06preventing.pdf.

[4] Abdalla, M., Bresson, E., Chevassut, O., Möller, B. and Pointcheval, D. (2007) Strong Password-based Authentication in TLS Using the Three-party Group Diffie-Hellman Protocol, *Int. J. Security and Networks, Vol. 2, Nos. 3/4, pp.284–296*. Retrieved May 31, 2015, from http://www.ssi.gouv.fr/uploads/IMG/pdf/AbdBreCheMol_07.pdf.

[5] Dhamija, R., Tygar, J. D., & Hearst, M. (2006, April). Why Phishing Works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems (pp. 581-590). ACM*. Retrieved, May 31, 2015, from http://escholarship.org/uc/item/9dd9v9vd.

[6] ITU-T X.1035: Password-Authenticated Key Exchange (PAK) protocol (2007). Retrieved May 31, 2015, from http://www.itu.int/rec/T-REC-X.1035-200702-I/en.

[7] Griffin, P. (2015, April). Formal Security Protocol Analysis. *Information Systems Security Association Journal*, Vol. 13, No. 4, April 2015.

About the Author

Phillip H. Griffin, CISM, has over 20 years experience in the development of commercial, national, and international security standards and cryptographic messaging protocols. Phil has a Master of Information Technology, Information Assurance and Security degree, and he has been awarded nine US patents at the intersection of biometrics, radio frequency identification (RFID), and information security management. He may be reached at phil@phillipgriffin.com.

Book Review

Robert Slade

Security for Service Oriented Architectures

Walt Williams

Auerbach Publications (2014)

Walt Williams is one of the sporadic, but thoughtful, posting members of the international CISSP Forum. He has come up with a significant text on an important topic. After some preface and introduction, the book starts in chapter two, defining the four kinds of architecture in computer systems: infrastructure, software, data, and security. This chapter covers foundational concepts as well as service-oriented architecture (SOA) and is, alone, worth the price of the book. Chapter three, on implementation, comprises the bulk of the space in the work and is primarily of interest to those dealing with development, although it does have a number of points and observations of use to the manager or security practitioner.

"Web 2.0" (chapter four) has some brief points on those advanced usages. A variety of additional SOA platforms are examined in chapter five. Chapter six, on the auditing of SOA applications, covers not only the how but also notes specific types of attacks and the most appropriate auditing tools for each case. Much the same is done, in terms of more general protection, in chapter seven. Chapter eight, simply entitled "Architecture," finishes off with sample cases.

It is an unfortunate truism that most security professionals do not know enough about programming, and most programmers don't care anything about security. This is nowhere truer than in service-oriented architecture and the cloud, where speed of release and bolt-on functionality trumps every other consideration. Williams' work is almost alone in a badly under-served field. Despite a lack of competition, it is a worthy introduction. I can recommend this book to anyone involved in either security or development, particularly those working in that nebulous concept known as the cloud."

About the Reviewer

Robert M. Slade, an information security and management consultant from North Vancouver, British Columbia, Canada. He may be reached at rmslade@shaw.ca.