Elliptic Curves in Transport Layer Security (TLS) - A Presentation Tutorial

Balakrishnan Vasudevan
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, California - 93106
balakrishnanvasudevan@umail.ucsb.edu

***Abstract*—Transport Layer Security (TLS) is a mechanism used to provide privacy and data security between two communicating applications. All major web browsers provide support for TLS to secure communications between them and the web servers. TLS ensures that the communication between the two applications is private using symmetric cryptography. Public Key Cryptography is optionally used to authenticate the identity of communicating devices. TLS also provides Message authentication to ensure reliable communication between the applications. This tutorial explains the TLS algorithm for key exchange, ciphering and message authentication. The various Elliptic Curve cryptographic functions being used in the current version TLS 1.2 and the draft version of TLS 1.3 are explained. It also discusses the strengths and vulnerabilities of algorithms like Elliptic Curve Diffie Hellman, Elliptic Curve Ephemeral Diffie Hellman and Elliptic Curve Digital Signature.**

I. Introduction

By the end of the year 2016, approximately 2.94 billion people and more than 25 billion devices [1] will have accessed 1.1 zetabytes of data that includes anything from movies on Netflix or papers on arXiv to grocery shopping and cash transfers on the internet. Securing these transactions is of utmost importance. In February 2015, it was discovered that the insurance provider Anthem was the target of an attack that resulted in more than 80 million records of Social Security numbers, email and physical addresses being stolen.

Public-key cryptographic systems use the trapdoor function to achieve security. The trapdoor is based on the fact that securing a message is easy, but it is a numerically hard problem to decrypt the message when a user does not have the private key. Elliptic curve cryptography systems make the trapdoor function even harder. To find the roots of an elliptic curve system is infeasible and is known as the Elliptic Curve Discrete Logarithmic Problem (ECDLP).

The security of an elliptic curve cryptosystem depends on the ability to compute a point multiplication and the inability of the attacker to to calculate the multiplicand when he is given the original and product points. The situation is akin to having a person play pool alone in a room. The person hits the cue ball to displace the other balls. When an another person with thorough knowledge of the game enters the room, it would be highly unlikely that the person can find the initial position of the ball given its current position.

Elliptic curves were suggested for usage by Neal Koblitz and Victor Miller independently in 1985. The NIST has since endorsed the Elliptic Curve Diffie Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm in the Suite B set of recommended algorithms.

TLS is highly flexible and lets developers use various schemes for encryption, authentication and key exchange. This will go a long way in ensuring the relevance of TLS in the current and future. The flexibility is also the main reasons why TLS is being considered to secure IoT applications which keep growing every single day.

The rest of the paper is organized as follows: *Transport Layer Security* details the fundamentals of the concept of TLS, *TLS Algorithm* explains the various steps followed in securing data from the higher layers, *Elliptic Curves in TLS* discusses the various Elliptic curves used in the current version TLS 1.2, the section *TLS 1.3* explains how the version of TLS currently in the pipeline is different from its predecessors. *Attacks and Mitigation* discusses attacks that have been carried out so far on TLS and steps for their mitigation.

II. Transport Layer Security

Transport Layer Security uses cryptographic systems to ensure privacy between a server and a client. Although the name suggests that the protocol operates in the Transport layer, TLS operates in the Application layer of the IP protocol suite. In the OSI model, TLS is initialized and operated from the Presentation and Session layers respectively.

The predecessors to TLS were Secure Network Programming and SSL (Secure Sockets Layer). SSL is still being used by a few websites. The earlier versions of TLS were TLS 1.0 and TLS 1.1 which have since been upgraded to TLS 1.2 with enhanced support for cipher suites and providing greater abilities to the client and the server to negotiate the hash and signature algorithms that are being used. With TLS 1.2, backward compatibility with SSL 2.0 has been discontinued [5].

TLS is made up of two protocols viz. TLS Record protocol, which is optional and which secures a connection using DES and the TLS Handshake protocol, which allows the client and the server to authenticate each other using a certificate provider and negotiate an encryption algorithm based on their computational capabilities to encrypt the data being exchanged between the two.

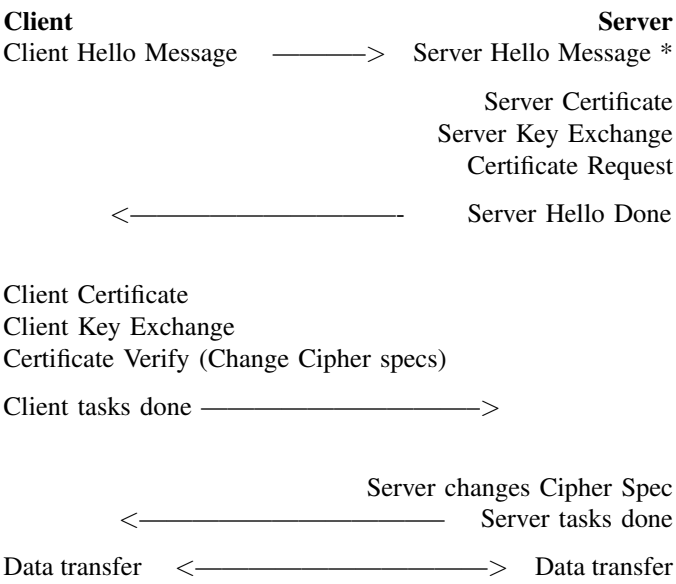
The current version of TLS, TLS 1.2 is defined by RFC 5246 [2].

III. TLS Algorithm

The following are the various steps involved in TLS [3],

1) The client requests a TLS enabled server for a secure connection and presents a list of cipher suites that it supports.
2) The server responds by picking up a cipher and a hash function and sends its digital certificate. The certificate has its name, the identity of the trusted certificate authority and its public encryption key.
3) The client can, if it so requires, contact the certificate authority to verify the authenticity of the server.
4) With the exchange of the certificates and other information, the session keys are generated. The client encrypts a random number with the server's public key which it then sends to the server. This random number can be decoded by the server with its private key. The random number is then used for encryption and decryption by the client and the server.
5) The client uses Diffie-Hellman key exchange to securely generate and transmit a random and unique session key for the purposes of encryption and decryption.
6) With this, the handshaking is complete and the secure connection is initiated. The session keys are used till the end of the session for encryption and decryption of data.

Failure of any of the above steps results in the TLS handshake being taken down.



A. Forward Secrecy

Forward secrecy is the property by which cryptosystems ensure that even if the server or the client's private key is disclosed in the future, the integrity of the session would not be compromised. Forward secrecy also prevents the data from being decrypted even if the session was recorded by an attacker.

Forward secrecy is achieved by computing a random key for each session. This ensures that if one key is compromised, it does not result in the loss of integrity of other keys that are generated later on for the session. Google has been providing forward secrecy with TLS for the Gmail and Google Docs service and Twitter has also been using TLS with forward secrecy for its services since November, 2013.

IV. Elliptic Curves in TLS

There has been a renewed interest in the use of elliptic curve cryptosystems given the level of security they provide for a smaller key size. TLS be made more secure by the use of ECC. RFC 4492 [4] specifies the list of elliptic curve systems being used in TLS. A comparison of the comparable key sizes for symmetric and asymmetric key cryptosystems was analyzed in [6] and is shown below,

Symmetric	ECC	DH/DSA/RSA
80	163	1024
112	233	2048
128	283	3072
192	409	7680
256	571	15360

The following elliptic curve cryptosystems are used in TLS 1.2,

1) ECDH + ECDSA
2) ECDHE + ECDSA
3) ECDH + RSA
4) ECDHE + RSA
5) ECDH + Anon

The first elliptic curve specifies the cryptosystem used to generate the public key and the second protocol specifies the system used to sign the certificate. For example, ECDH + ECDSA uses an Elliptic Curve Diffie Hellman scheme to generate a public key. The client then generates an ECDH key pair on the same curve as the server's long term public key and then sends its public key in the Client Exchange message that it sends to the server.

ECDHE refers to Elliptic Curve Diffie Hellman Ephemeral meaning that the key is generated for each and every execution of the key establishment process. ECDH + RSA and ECDHE + RSA are used to let the server can reuse its existing RSA certificate. This scheme does not provide forward security which is the key feature of TLS.

ECDH + anon does not provide any authentication of the server or the client. The server and the client's certificate and the certificate request and the certificate verify messages are sent in order to ensure anonymity. The server uses the ECDH algorithm to generate the public key and sent to the client. The Server Key Exchange message is not signed by the server. When the client receives the message, it generates a curve pair from the same curve and sends the public key back to the server in the Client Key Exchange message. This scheme is prone to man-in-the-middle attacks due to the lack of authentication.

TLS 1.2 also proposed two new extensions for use of ECC in TLS,

1) Supported Elliptic curves extension
2) Supported Points format extension

Both these schemes allow the system to negotiate the use of specific points and curves during the handshaking phase. Both the client and the server should support the above schemes so as to be able to use them during data transfer.

V. TLS 1.3

The draft specification for TLS 1.3 [7] is still being worked upon. The following are some of the major changes being proposed in the new version,

1) Support for weak and lesser used Elliptic curves like ECDH+RSA and ECDH+ECDSA is being withdrawn.
2) Support for hash functions like MD5 and SHA-224 is being withdrawn.
3) Requirement for a digital signature even when a previous configuration is being used.
4) Use of SSL and RC4 negotiation for backward compatibility is being withdrawn.
5) Authentication modes unified and post-handshake client authentication has been added.
6) Client and server key shares have been merged into a single extension.
7) Support for DSA has been removed and MTI algorithms have been added.

VI. Attacks and Mitigation

RFC 7457 details the known attacks on TLS and datagram TLS [8]. A *version rollback attack* is carried out by influencing the cipher suite strength to a weaker symmetric key exchange algorithm which the attacker can easily break in to. *BEAST* or *Browser Exploit against SSL/TLS* uses vulnerabilities in TLS 1.0 implementations to decrypt HTTP cookies when HTTP is run over TLS. Attacks like *TIME* or *BREACH* use HTTP level compression to decrypt data being passed in the HTTP response. TIME can be mitigated by disabling compression in HTTP, however no known schemes of mitigation exist for protection against a BREACH attack. major vulnerabilities have also been found in the RC4 cryptosystem with the keystream being used to recover repeatedly encrypted plaintext. Current versions of most browsers are capable of mitigating most of these attacks.

For Elliptic Curves in TLS, attacks can usually exploit the structure of the elliptic curves being used. It is conservative to use elliptic curves for TLS with as little algorithmic structure as possible. Given this condition, random curves are more conservative than Koblitz curves. Another issue with the use of Elliptic Curves in TLS is the repeated use of a single elliptic curve. In this scenario, an attack can result in a large number of keys being compromised.

Forward secrecy also plays a major role in selecting elliptic curve protocols for use in TLS. Forward secrecy ensures that the keys securing a session are not compromised in the event that the certified keys from the client or the server are compromised. ECDHE+ECDSA and ECDHE+RSA provide forward secrecy in the event that the server's keys are compromised. use of ECDH+ECDSA and ECDH+RSA offer no protection in the event that the server's key has been compromised, similarly ECDHA+Fixed ECDH and RSA+Fixed ECDH offer no protection if the client's keys have been compromised.

Both ECDHE+ECDHSA and ECDHE+RSA schemes offer forward secrecy, ensuring secrecy even the server or the client's keys have been compromised. The list of curves approved by the various standards organizations has been provided in Table I.

TABLE I
List of Approved Curves

SECG	ANSI X9.62	NIST
sect163k1	-	K-163
sect163r1	-	-
sect163r2	-	B-163
sect193r1	-	-
sect193r2	-	-
sect233k1	-	K-233
sect233r1	-	B-233
sect239k1	-	-
sect283k1	-	K-283
sect283r1	-	B-283
sect409k1	-	K-409
sect409r1	-	B-409
sect571k1	-	K-571
sect571r1	-	B-571
secp160k1	-	-
secp160r1	-	-
secp160r2	-	-
secp192k1	-	-
secp192r1	prime192v1	P-192
secp224k1	-	-
secp224r1	-	P-224
secp256k1	-	-
secp256r1	prime256v1	P-256
secp384r1	-	P-384
secp521r1	-	P-521

VII. Conclusion

The ever increasing number of services offered on the internet, the data deluge and the requirements for security for most of the applications is proof that TLS has to be capable of thwarting dynamically. A persistent issue with implementing a strong TLS is the reluctance of websites and clients to upgrade their set of cryptographic suites to better versions. Complex cryptographic algoithms put a greater load on devices thereby causing a significant drain of power and memory resources. Internet of Things (IoT) is growing to encompass major aspects of our life currently and ensuring the security of these applications and devices is of utmost importance. Use of Elliptic Curves can go a long way in securing TLS given the infallibility of the curves to most kinds of attacks. However, usage of complex elliptic curves comes with its own share of problems. Complex curves are difficult to implement and compute and would not be suitable for devices with limited computational capabilities. Similarly use of algorithms like ECDHE+ECDSA for forward secrecy would also result in implementation difficulties. An ideal cryptosystem would have to harness the potential of Elliptic Curves without placing significant limitations on the implementation or the powers of computation.

Acknowledgment

I would like to thank Professor Dr. Çetin Kaya Koç with the Computer Science department of the University of California, Santa Barbara for suggesting the topic and for his guidance during the course of the project.

REFERENCES

[1] Cisco Systems Inc. *The Zettabyte Era-Trends and Analysis - Cisco Visual Networking index*.

[2] T. Dierks, E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246, IETF, August 2008.

[3] Wikipedia article on Transport Layer Security. https://en.wikipedia.org/wiki/Transport_Layer_Security.

[4] S. Blake-Wilson et al. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*. RFC 4492, IETF, May 2006.

[5] Turner and Polk. Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176, IETF Network Working Group, March 2011

[6] Lenstra, A. and E. Verheul, *Selecting Cryptographic Key Sizes*. Journal of Cryptology 14 (2001) 255-293

[7] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Draft, IETF Network Working Group, December 2015

[8] Y. Sheffer et al. *Summarizing Known Attacks on Transport Layer Security (TLS)and Datagram TLS (DTLS)*. RFC 7457, IETF, Feb 2015.