CS558 Network Security

15

Lecture ~~14~~: Finishing up TLS

BOSTON UNIVERSITY

TLS(1.2) in Detail

RFC 5246 TLS August 2008

negotiated

Client	Server
ClientHello --------->
	ServerHello
	Certificate*
	ServerKeyExchange*
	CertificateRequest*
	<--------- ServerHelloDone
~~Certificate*~~
ClientKeyExchange
~~CertificateVerify*~~
[ChangeCipherSpec]
Finished --------->
	[ChangeCipherSpec]
	<--------- Finished
Application Data <-------->	Application Data

negotiate authenticate confidentially

$Enc_{pk}(Pms)$

g^a

H(messages)

H(msg)

Figure 1. Message flow for a full handshake

BOSTON UNIVERSITY

Figure 1 below shows the basic full TLS handshake:

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^  Server
                                        {CertificateRequest*}  v  Params
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

g^a

BOSTON UNIVERSITY

TLS1.3

Break backwards Compatibility

- Get rid of old Crypto (eg. Export Grade Crypto)
- Get rid of MAC then Encrypt (AEAD by default)
- Perfect Forward Secrecy → only Support EDH
- Prevent Downgrades (Change Signing)
- Reduce Latency (Remove Roundtrips)
- 0-RTT

Authenticated Encryption with Associated Data

AES_GCM

MAC [Enc(plaintext) || AD]

TLS1.3 Cipher Suites

a

g

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

Perfect Forward Secrecy

hello

bad hello

Verify(, , σ) = {0,1}

g^b , , , σ

Enc(H(msg) || MAC)

TLS 1.3

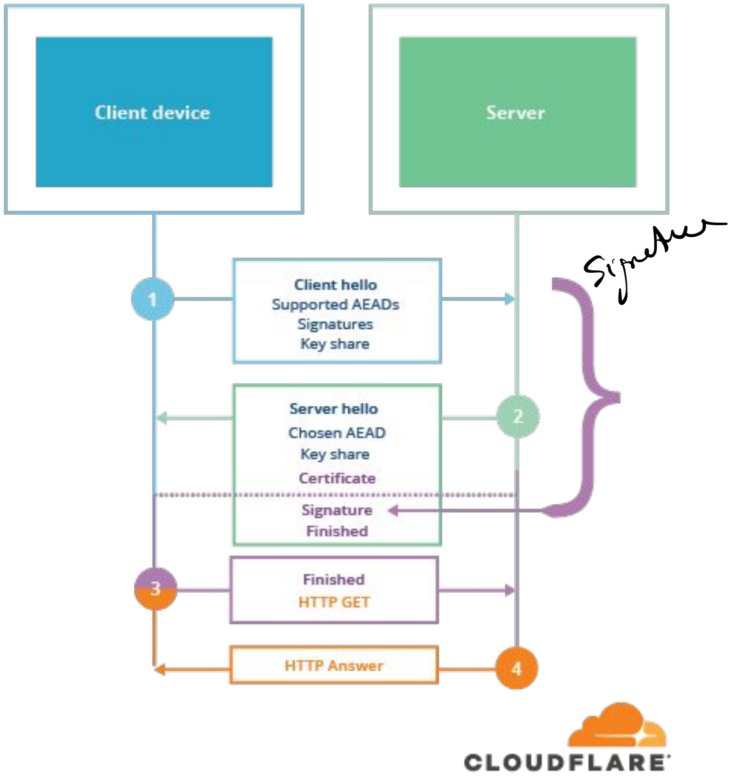
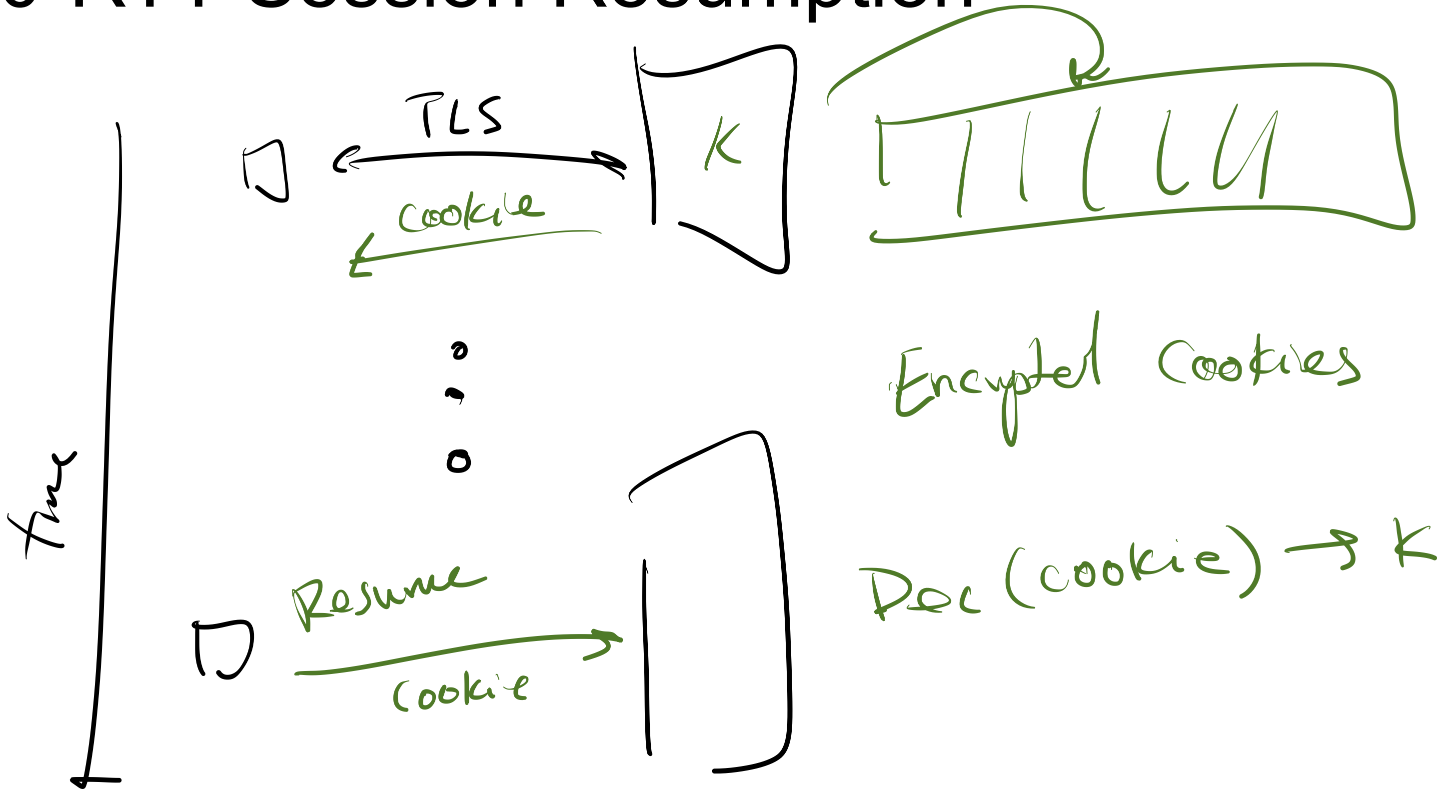

Client device
Server
Client hello
Supported AEADs
Signatures
Key share
Server hello
Chosen AEAD
Key share
Certificate
Signature
Finished
Finished
HTTP GET
HTTP Answer
1
2
3
4
Signature
CLOUDFLARE

Figure 1 below shows the basic full TLS handshake:

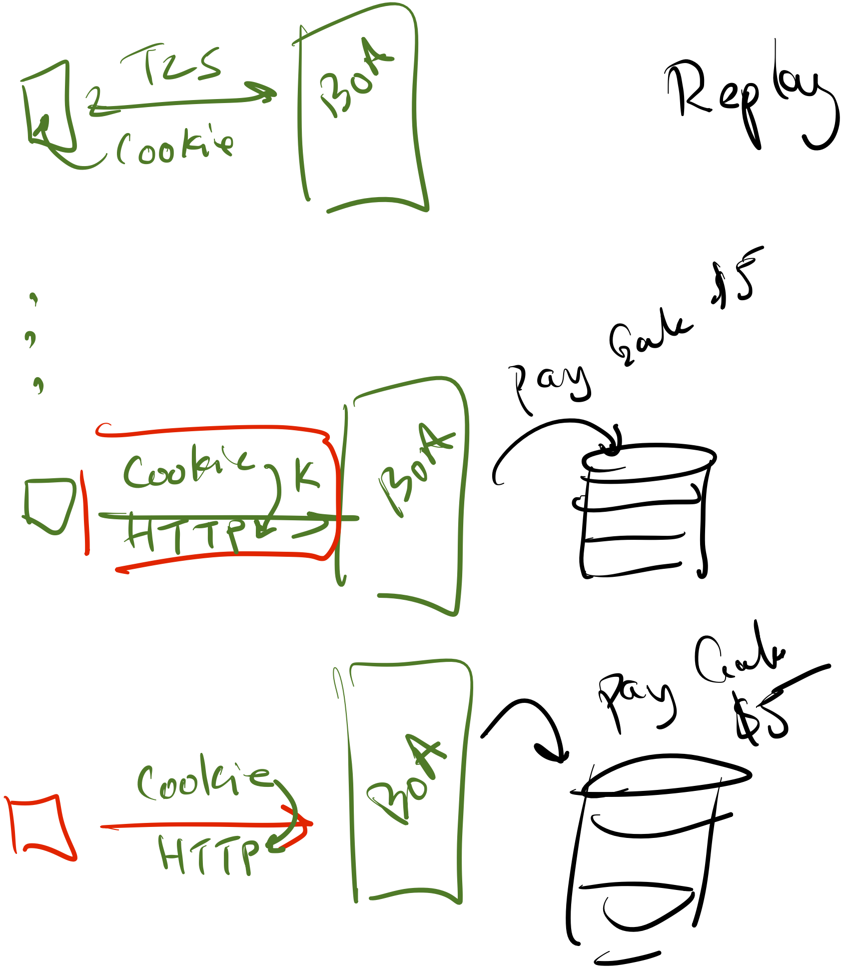
```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*     g^a
     v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch  g^b
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^  Server
                                        {CertificateRequest*}  v  Params
                             Encrypted         {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

0-RTT Session Resumption



Replay attack



If negotiating TLS 1.2, TLS 1.3 servers MUST set the last 8 bytes of
their Random value to the bytes:

```
  44 4F 57 4E 47 52 44 01
```

A Comprehensive Symbolic Analysis of TLS 1.3

Cas Cremers
University of Oxford, UK

Marko Horvat
MPI-SWS, Germany

Jonathan Hoyland
Royal Holloway, University of London, UK

Sam Scott
Royal Holloway, University of London, UK

Thyla van der Merwe
Royal Holloway, University of London, UK

ABSTRACT

The TLS protocol is intended to enable secure end-to-end communication over insecure networks, including the Internet. Unfortunately, this goal has been thwarted a number of times throughout the protocol's tumultuous lifetime, resulting in the need for a new version of the protocol, namely TLS 1.3. Over the past three years, in an unprecedented joint design effort with the academic community, the TLS Working Group has been working tirelessly to enhance the security of TLS.

We further this effort by constructing the most comprehensive, faithful, and modular symbolic model of the TLS 1.3 draft 21 release candidate, and use the TAMARIN prover to verify the claimed TLS 1.3 security requirements, as laid out in draft 21 of the specification. In particular, our model covers *all* handshake modes of TLS 1.3.

Our analysis reveals an unexpected behaviour, which we expect will inhibit strong authentication guarantees in some implementations of the protocol. In contrast to previous models, we provide a novel way of making the relation between the TLS specification and our model explicit: we provide a fully annotated version of the specification that clarifies what protocol elements we modelled, and precisely how we modelled these elements. We anticipate this model artifact to be of great benefit to the academic community and the TLS Working Group alike.

KEYWORDS

symbolic verification, authenticated key exchange, TLS 1.3

1 INTRODUCTION

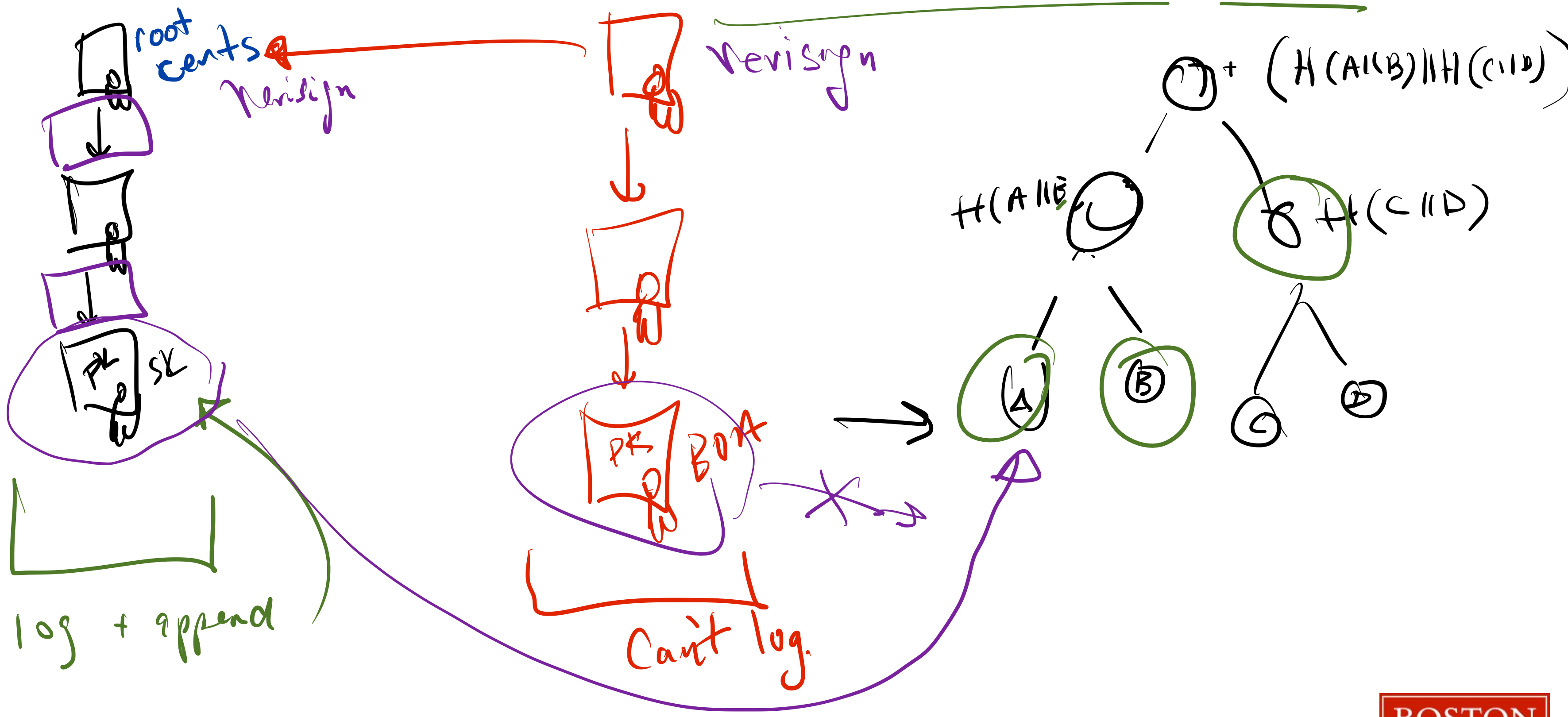
The Transport Layer Security (TLS) protocol is the *de facto* means for securing communications on the World Wide Web. Initially released as Secure Sockets Layer (SSL) by Netscape Communications in 1995, the protocol has been subject to a number of version upgrades over the course of its 20-year lifespan. Rebranded as TLS when it fell under the auspices of the Internet Engineering Task Force (IETF) in the mid-nineties, the protocol has been incrementally modified and extended. In the case of TLS 1.2 and below, these modifications have taken place in a largely retroactive fashion; following the announcement of an attack [6, 7, 18, 20, 32, 43, 49], the TLS Working Group (WG) would either respond by releasing a protocol extension (A Request for Comments (RFC) intended to provide increased functionality and/or security enhancements) or by applying the appropriate "patch" to the next version of the protocol. For a more detailed analysis of the development and standardisation of TLS see [45].

Prior to the announcement of the BEAST [26] and CRIME [27] attacks of 2011 and 2012, respectively, such a strategy was valid given the frequency with which versions were updated, and the limited number of practical attacks against the protocol.

Post-2011, however, the heightened interest in the protocol and the resulting flood of increasingly practical attacks against it [1–3, 5, 9, 13, 15, 16, 26, 27, 29, 31, 41, 42, 44] rendered this design philosophy inadequate. Coupled with pressure to increase the protocol's efficiency (owing to the release of Google's QUIC Crypto [37]), the IETF started drafting the next version of the protocol, TLS 1.3, in the Spring of 2014. Unlike the development of TLS 1.2 and below, the TLS WG adopted an "analysis-prior-to-deployment" design philosophy, welcoming contributions from the academic community before official release. There have been substantial efforts from the academic community in the areas of program verification– analysing implementations of TLS [12, 14], the development of computational models– analysing TLS within Bellare-Rogaway style frameworks [24, 25, 28, 33, 35, 38], and the use of formal methods tools such as ProVerif[17] and Tamarin[48] to analyse symbolic models of TLS [4, 10, 22, 30]. All of these endeavours have helped to both find weaknesses in the protocol and confirm and guide the design decisions of the TLS WG.

The TLS 1.3 draft specification however, has been a rapidly moving target, with large changes being effected in a fairly regular fashion. This has often rendered much of the analysis work 'outdated' within the space of few months as large changes to the specification effectively result in a new protocol, requiring a new wave of analysis.

New Ideas Are Still Needed -- Certificate Transparency Logs



TLS 1.2

$Enc(H(m) \| MAC)$

H(m) is Secure iff
the Shared key is
Secure

TLS 1.3

Transcript

T, σ

T is untampered-with
as long as the
Server's Secret
Key is Secure

BOSTON UNIVERSITY

Post-quantum key exchange – a new hope*

$g^a \longrightarrow a$

Erdem Alkim
Department of Mathemathics, Ege University, Turkey

Léo Ducas
Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands

Thomas Pöppelmann
Infineon Technologies AG, Munich, Germany

Peter Schwabe
Digital Security Group, Radboud University, The Netherlands

Abstract

In 2015, Bos, Costello, Naehrig, and Stebila (IEEE Security & Privacy 2015) proposed an instantiation of Ding's[1] ring-learning-with-errors (Ring-LWE) based key-exchange protocol (also including the tweaks proposed by Peikert from PQCrypto 2014), together with an implementation integrated into OpenSSL, with the affirmed goal of providing post-quantum security for TLS.

1 Introduction

The last decade in cryptography has seen the birth of numerous constructions of cryptosystems based on lattice problems, achieving functionalities that were previously unreachable (e.g., fully homomorphic cryptography [43]). But even for the simplest tasks in asymmetric cryptography, namely public-key encryption, signatures, and key exchange, lattice-based cryptography offers an

PQC Standardization Process: Third Round Candidate Announcement

July 22, 2020

It has been almost a year and a half since the second round of the NIST PQC Standardization Process began. After careful consideration, NIST would like to announce the candidates that will be moving on to the third round. The seven third-round Finalists are:

Third Round Finalists

Public-Key Encryption/KEMs
Classic McEliece
CRYSTALS-KYBER
NTRU
SABER

Key Encapsulation mechanism

$Enc_{pk}(pms)$

Digital Signatures
CRYSTALS-DILITHIUM
FALCON
Rainbow

PARENT PROJECT

See: Post-Quantum Cryptography

RELATED TOPICS

Security and Privacy: digital signatures, key management, post-quantum cryptography

Activities and Products: standards development

RELATED PAGES

Event: Third PQC Standardization Conference

Selected Algorithms 2022

Official comments on the Selected Algorithms should be submitted using the "Submit Comment" link for the appropriate algorithm. Comments from the pqc-forum Google group subscribers will also be forwarded to the pqc-forum Google group list. We will periodically post and update the comments received to the appropriate algorithm.

All relevant comments will be posted in their entirety and should not include PII information in the body of the email message.

Please refrain from using OFFICIAL COMMENT to ask administrative questions, which should be sent to pqc-comments@nist.gov

History of Selected Algorithms Updates

Selected Algorithms: Public-key Encryption and Key-establishment Algorithms

Algorithm	Algorithm Information	Submitters	Comments
CRYSTALS-KYBER **PQC License Summary & Excerpts**	Zip File (7MB) IP Statements Website	Peter Schwabe Roberto Avanzi Joppe Bos Leo Ducas Eike Kiltz Tancrede Lepoint Vadim Lyubashevsky John M. Schanck Gregor Seiler Damien Stehle Jintai Ding	Submit Comment View Comments

Selected Algorithms: Digital Signature Algorithms

Algorithm	Algorithm Information	Submitters	Comments
CRYSTALS-DILITHIUM	Zip File (11MB) IP Statements Website	Vadim Lyubashevsky Leo Ducas Eike Kiltz Tancrede Lepoint Peter Schwabe Gregor Seiler Damien Stehle Shi Bai	Submit Comment View Comments
FALCON	Zip File (4MB) IP Statements Website	Thomas Prest Pierre-Alain Fouque Jeffrey Hoffstein Paul Kirchner Vadim Lyubashevsky Thomas Pornin Thomas Ricosset Gregor Seiler William Whyte Zhenfei Zhang	Submit Comment View Comments
SPHINCS+	Zip File (230MB) IP Statements Website	Andreas Hulsing Daniel J. Bernstein Christoph Dobraunig Maria Eichlseder Scott Fluhrer Stefan-Lukas Gazdag Panos Kampanakis Stefan Kolbl Tanja Lange Martin M Lauridsen Florian Mendel Ruben Niederhagen Christian Rechberger	Submit Comment View Comments

BOSTON UNIVERSITY

Where we go from here: Censorship Circumvention and E2E Messaging

- We know how to create a secure communication with the server
- What if someone doesn’t let us talk to the server?
- How much do we need to trust the server

BOSTON UNIVERSITY