

Microsoft Research - Inria JOINT CENTRE



Verified Secure Implementations for the HTTPS Ecosystem miTLS & Everest*

The HTTPS Ecosystem is critical



- Most widely deployed security?
 ¹/₂ Internet traffic (+40%/year)
- Web, cloud, email, VoIP, 802.1x, VPNs, ...

The HTTPS Ecosystem is complex



The HTTPS Ecosystem is broken

• 20 years of attacks & fixes Buffer overflows Incorrect state machines Lax certificate parsing Weak or poorly implemented crypto Side channels

Informal security goals Dangerous APIs Flawed standards

• Mainstream implementations OpenSSL, SChannel, NSS, ... Still patched every month!



Goal: a secure channel



Security Goal: As long as the adversary does not control the long-term credentials of the client and server, it cannot

- Inject forged data into the stream (authenticity)
- Distinguish the data stream from random bytes (confidentiality)

TLS protocol overview



Many configurations (some of them broken)



miTLS (2013—...) a first verified reference implementation

1. Internet Standard compliance & interoperability supporting SSL 3.0—TLS 1.2

2. Verified security:

we structured our code to enable its modular cryptographic verification, from its main API down to concrete algorithms (RSA, AES,...)

3. Experimental platform:

for testing corner cases, trying out attacks, analysing extensions and patches, ...

Excluding core crypto algorithms

Not fully automated (paper proofs too)

Not production code (poor performance)



Search GitHub



miTLS

A verified reference TLS implementation

○ http://www.mitls.org/

📮 Repositories

🕀 People 🕕

Filters - Q. Find a repository...

miTLS v0.9 released in Nov'15 https://github.com/mitls

mitls-fstar

TLS implemented in F*

Updated 17 hours ago

using F* (in progress) with early support for TLS 1.3

mitls-flex

TLS implemented in f7

Updated 14 days ago

using F# & F7 (stable) including testing tools

Verified Communications Security?



Application security (API, configuration) Cryptographic schemes & assumptions Protocol design Implementation safety Information control (leakage, privacy) Verification tools (F#, F7, F*, Z3, Lean) (1) data streams

- (2) main theorem
- (3) state-machine attacks

Modelling Secure Data Streams (1/2) Type abstraction for integrity & confidentiality

Ideally, TLS passes around data fragments; it cannot forge them, or read their contents.

Concretely, this reduces to probabilistic poly-time security assumptions on the underlying cryptographic primitives (e.g. INT-CCA + IND-CPA)

We use **type indexes** to separate between different streams, keep track of their lengths, and control coercions to concrete bytes // F* definition of Application Data

```
abstract type data (i:id) = bytes
```

let ghost #(i:id) (d:data i): GTot bytes = d

```
type fragment (i:id) (rg:range) =
    d:data i {within (ghost d) rg}
```

```
\rightarrow Tot (b:bytes {b = ghost d})
```

Modelling Secure Data Streams (2/2) Stateful invariant for stream authentication



Security Theorem

Main crypto result: concrete TLS & ideal TLS are computationally indistinguishable

Bytes, Network lib.fs Cryptographic Provider application cryptographic assumptions data stream miTLS ideal miTLS implementation implementation miTLS typed API miTLS typed API any program application representing the adversary Safe, except for a Safe by typing negligible probability (info-theoretically)

Verification technique: security by typing

Security Theorem

Proof automation 7,000 lines of F# verified against 3,000 lines of F7 type annotations

The security statement is precise but complex, roughly the size of the TLS API and cryptographic assumptions



Scripting Tools & Security Testing

miTLS clean, modular implementation supports rapid prototyping against others

- One line of F# script for each TLS message, with good cryptographic defaults
- Simple setup for "man-in-the-middle" attacks and concurrent connections
- Built-in library of recent vulnerabilities
- Fuzzing on the TLS state machine

Focus on ease of use (but still for experts)

Triple handshake attack (2014)

flaw in the standard now patched in TLS





https://www.secure-resumption.com/

Systematically testing the TLS state machine new attacks against all mainstream implementations

TLS offers many ciphersuites, optional messages, extensions... sharing the same state machine.

miTLS provides a verified TLS state machine.

We systematically generate and test deviant traces against other implementation (skipping, inserting, reordering valid messages)

We found many many exploitable bugs



Systematically testing the TLS state machine

new attacks against all mainstream implementations

TLS offers many ciphersuites, optional messages, extensions... sharing the same state machine.

miTLS provides a verified TLS state machine.

We systematically generate and test deviant traces against other implementation (skipping, inserting, reordering valid messages)



An attack against TLS Java Library (open for 10 years)

We skip 6 messages

JSSE's client assumes the key exchange is finished, uses uninitialized 0x000000... as session key!

FREAK: downgrade to RSA_EXPORT (2015)

Man-in-the-middle attack against:

- servers that support RSA_EXPORT (512bit keys obsoleted in 2000) from 40% to 8.5%
- clients that accept ServerKeyExchange in RSA (state machine bug) almost all browsers have been patched



Similar attack, different crypto: LOGJAM (2015) downgrade to weak groups

FREAK in the news



The Washington Post

'FREAK' flaw undermines security for Apple and Google users, researchers discover

Technology

Millions at risk from 'Freak' encryption bug

3 6 March 2015 | Technology

The New York Times

Apple, Android Browsers Vulnerable to 'FREAK Attack'

By THE ASSOCIATED PRESS MARCH 3, 2015, 9:06 P.M. E.S.T.



Technology

Apple and Google 'FREAK attack' leaves millions of users vulnerable to hackers



The .

Economist w

World politics Business & finance Economics Science & technology Cultur

Computer security

The law and unintended consequences

The perils of deliberately sabotaging security





COMPUTERS are notoriously insecure. Usually, this is by accident rather than design. Modern operating systems contain millions of lines of code, with millions more in the applications that do the things people want done. Human brains are simply too puny to build something so complicated without making mistakes.

On March 3rd, though, a group of researchers at Microsoft, an American computer company, Imdea, a Spanish research institute, and the National Institute for Research in Computer Science and Automation, in France, discovered something slightly different. They found a serious flaw in cryptography designed to guard private data such as e-mails,

MUST READ: Pandora buys Rdio for \$75 million





'Logjam' browser vulnerability fix will block thousands of websites

🧏 by Steve Dent | @stevetdent | May 20th 2015 At 9:45am

HTTPS-crippling attack threatens tens of thousands of Web and mail servers

Diffie-Hellman downgrade weakness allows attackers to intercept encrypted data.

by Dan Goodin - May 20, 2015 6:54am BST

MAY 27, 2015 | BY JOSEPH BONNEAU

ELECTRONIC FRONTIER FOUNDATION DEFENDING YOUR RIGHTS IN THE DIGITAL WORLD

Logjam, Part 1: Why the Internet is Broken Again (an Explainer)

TECH THE WALL STREET JOURNAL. New Computer Bug Exposes Broad Security Flaws

Fix for LogJam bug could make more than 20,000 websites unreachable

By JENNIFER VALENTINO-DEVRIES

May 19, 2015 7:02 p.m. ET



Q EDIT

Y f

Pwnie for Most Innovative Research

Awarded to the person who published the most interestin presentation, tool or even a mailing list post.

 Imperfect Forward Secrecy: How Diffie-He Credit: David Adrian et al.

This paper introduces the Logjam attack, a vulne downgrade TLS connections to 512-bit export-gra We found & fixed flaws in legacy implementations of TLS... probably many others still in there. Can we be more constructive?

Can we make the next TLS better?

- We are trying to model, implement, and improve the new draft standard (TLS 1.3)
- Deployment will take years—are TLS 1.2 and TLS 1.3 jointly secure?

Can we deploy verified code in the TLS/HTTPS ecosystem?

- Despite great technical achievements, formally verified software is seldom deployed and used
- TLS is small & critical, can be exemplary case for verified deployed software

TLS 1.3: a new hope

Much discussions

IETF, Google, Mozilla, Microsoft, CDNs, cryptographers, network engineers, ...

Much improvements

- Modern design
- Fewer roundtrips
- Stronger security

New implementations required for all

- Be first & verified too!
- Find & fix flaws before it's too late

Network Working Group Internet-Draft Obsoletes: 5077, 5246, 5746 (if approved) Updates: 4492 (if approved) Intended status: Standards Track Expires: September 23, 2016

The Transport Layer Security (TLS) Protocol Version 1.3

draft-ietf-tls-tls13-latest

Abstract

This document specifies Version 1.3 of the Transport Layer Security (TLS) protocol. The TLS protocol allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Status of This Memo

This Internet Droft is au		5.1.	Connection States									
79.	🖫 tlswg / tls13-spec		• Watch	54	★ Star	130	∜ Fork	57				
Internet-Drafts are work that other groups may a current Internet-Drafts i	⇔ Code ① Issues 32 11 Pull requests 9 11 Wiki → Pulse III Graphs											
Internet-Drafts are draft	Filters • Q is:issue is:open Labels Milestones						New is	ssue				
Drafts as reference ma	① 32 Open ✓ 155 Closed	uthor 🕶	Labels -	Milest	ones 🕶	Assignee	e≖ So	ort +				
This Internet-Draft will (Define what the SignatureScheme code points are #441 opport 3 hours are by marticiphorese				Ç 0							
Copyright N												
Copyright (c) 2016 IETI reserved.	O PKCS1 #440 opened 3 hours ago by martinthomson erved.											
This document is subje Documents (http://trusti document. Please revie restrictions with respec must include Simplified Provisions and are prov This document may cor or made publicly availa copyright in some of thi modifications of such m adequate license from 1 document may not be r it may not be created o as an RFC or to transle	O-RTT when the server rejects a ClientHello H438 opened 14 days ago by martinthomson											
	With resumption PSK, make the PSK label partly derived from the session hash #427 opened on Mar 9 by ekr											
	Remove DH-based 0-RTT #425 opened on Feb 23 by martinithomson											
	① Add encrypted NextRecordLength field to make next record's unencrypted he #422 opened on Feb 23 by bford	ader optio	onal parked					ÇI 1				
	③ PSK and Certificates? #421 opened on Feb 21 by wbl							Ç 0				
	Remove client authentication from 0-RTT #420 opened on Feb 21 by martinthomson							Ç 1				
1. Introducti	Should EncryptedExtensions have an inner list #419 opened on Feb 21 by ekr							Ç 0				
DISCLAIMER: This is a analysis.	① Have the server provide the PSK index not the label? #418 opened on Feb 21 by ekr							Γ Ο				
RFC EDITOR: PLEASE	Allow servers to send KnownGroups											

Table of Contents

- 1.1. Conventions and Terminology
- 1.2. Major Differences from TLS 1.2
- 2. Goals

E. Rescorla

RTFM. Inc.

March 22, 2016

- 3. Goals of This Document
- 4. Presentation Language
- 4.1. Basic Block Size 4.2. Miscellaneous
- 4.3. Vectors
- 4.4. Numbers

4.5. Enumerateds 4.6. Constructed Types

4.6.1. Variants

5. The TLS Record Protocol

- 4.7. Constants
- 4.8. Cryptographic Attributes
- 4.8.1. Digital Signing4.8.2. Authenticated Encryption with Additional Data (AEAD)

TLS 1.3: status

IETF TLS WG95 (April'16)

• 13th draft discussed

Adopting several of our proposals: extended session hashes, downgrade resilience, pre-shared-key ORTT, session ticket format, simplified key schedule...

• Finalized in 6 months?

	Network Working Group Internet-Draft Obsoletes: 5077, 5246, 57 approved) Updates: 4492 (if approved) Intended status: Standards Expires: September 23, 20 The Transp Protocol Ve	r Secu 3	ırity (T	E. Rescorta RTFM, Inc. March 22, 2016	Table 1. Introc 1.1. 2. Goak 3. Goak 4. Prese 4.1. 4.2. 4.3.	Die of Contents Iroduction 1. Conventions and Terminology 2. Major Differences from TLS 1.2 bals bals of This Document esentation Language 1. Basic Block Size 2. Miscellaneous 3. Vectors			
	Imp	lementat	tion St	tatus			imbers numerateds instructed Types 1. Variants instants yptographic Attributes		
Name	Language	ECDHE	DHE	PSK	0-RTT		.1. Digital Signing 2. Authenticated Encryption with		
NSS	С	Yes	No	Yes	Yes*		Additional Data (AEAD) S Record Protocol		
Mint	Go	Yes	Yes	Yes	Yes		onnection States > Watch - 54 ★ Star 130 ¥		
nqsb	OCaml	No	Yes	Yes	No				
ProtoTLS	JavaScript	Yes	Yes	Yes	Yes		N		
miTLS	F*	Yes	Yes	Yes	???		bels ▼ Milestones ▼ Assignee ▼		
• NSS inte	erops with Mi	int and Pro	otoTLS						

ProtoTLS interops with ngsb

- NSS 0-RTT in unintegrated branch

Other combinations untested

IETF

TLS		
PSK and Certificat #421 opened on Feb 21	adequate license from t document may not be r	
Remove client aut #420 opened on Feb 21	it may not be created o as an RFC or to transla	
Should Encrypted #419 opened on Feb 21	1. Introducti	
Have the server pr #418 opened on Feb 21	DISCLAIMER: This is a analysis.	
O Allew 4	RFC EDITOR: PLEASE	

Allow servers to server

Abstract—Key-exchange protocols such as TLS, SSH, IPsee and ZRTP are highly configurable, with typical deployment upporting multiple protocol versions, cryptographic algorithm and parameters. In the first messages of the protocol, the peer regotiate one specific combination: the protocol mode, based on heir local configurations. With few notable exceptions, mos
ryptographic analyses of configurable protocols consider a single node at a time. In contrast, downgrade attacks, where a single
adversary forces peers to use a mode to a standard a
How to To anneal a
Juestion by O a Provide to study downgrade
esilience a second second to other security properties of key exchange protocols. First, we study the causes of downgrade
attacks by dissecting and classifying known and novel attacks against widely used protocols. Second, we survey what is known

Downgrade Resilience in Key-Exchange Protocols Karthikeyan Bhargavan*, Christina Brzuska[†], Cédric Fournet[‡], Matthew Green[§],

> Markulf Kohlweiss[‡] and Santiago Zanella-Béguelin[‡] *Inria Paris-Rocquencourt, Email: karthikeyan.bhargavan@inria.fr Hamburg University of Technology, Email: brzuska@tuhh.de [‡]Microsoft Research, Email: {fournet,markulf,santiago}@microsoft.com §Johns Hopkins University, Email: mgreen@cs.jhu.de



¥ Fork 57

e - Sort -

0 💭

Fig. 1: SIGMA-N: Basic SIGMA [30] with group negotiation

Everest (2016—2021): Verified Drop-In Replacements for the HTTPS ecosystem



Everest Goals

- Strong verified security
- Widespread deployment
- Trustworthy usable tools







Application Security: https://

Demo: tracing https://www.visualstudio.com/

Trust is transitive

each page involves connections to many servers (different origins)

• Trust is implicit 17 concurrent TLS connections,

configurations, certificate chains

• Trust is a matter of state cookies, caches, configurations, proxies

www.visualstudio.com - F12 Developer Tools - Microsoft Ed F12 DOM Explorer Console Debugger	ge Network	Per	formance	Memory Emulation	Experiments			×
🕨 📕 🕍 🚰 🍆 🐌 🎽 🔽	ontent typ	pe						Find (Ctrl+F)
Name / Path	Protoc	Method	Result / Description	Content type	Received	Time	Initiator / Headers Body Parameters	Cookies Timings
https://www.visualstudio.com/	HTTPS	GET	200 OK	text/html	17.45 KB	655.72 ms	document Request URL: https://www.visua	lstudio.com/
wt.js https://cwebtrends.com/acs/account/iBowcm6p7//is/	HTTPS	GET	200 OK	application/javascript	10.68 KB	48.46 ms	script Status Code: 200 / OK	
Combined.css?resources=0:Layout,0:ImageSprite,0:BGCol	HTTPS	GET	200	text/css	9.77 KB	19.32 ms	link A Request Headers	html.coml image/out #/#
Combined.css?resources=0:Layout,0:ImageSprite,0:BGCol	HTTPS	GET	200	text/css	(from cache)	0 s	Accept-Encoding: gzip, deflate,	peerdist
sizzle_1.min.js	HTTPS	GET	200	application/javascript	6.64 KB	11.86 ms	script Accept-Language: en-GB, en; q	=0.8, fr-FR; q=0.5, fr; q=0.3
optimizejs	HTTPS	GET	200	application/javascript	20.13 KB	24.86 ms	script Cookie: ASPSESSIONIDSSDBSBF	RS=LKKBNICDNCAMGGG.
Loader.js	HTTPS	GET	304	application/javascript	(from cache)	20.27 ms	script Host: www.visualstudio.com	
nttps://i2-vso.sec.s-mstt.com/Areas/Global/Content/ Combined.js?resources=0:Utilities,1:FixUnevenHeights,2:L	HTTPS	GET	200	application/javascript	8.6 KB	15.66 ms	User-Agent: Mozilla/5.0 (Windo script X-P2P-PeerDist: Version=1.1	ws NT 10.0; Win64; x64)
https://i2-vso.sec.s-msft.com/ SearchBox.jss?boxid=HeaderSearchTextBox&btnid=Head	HTTPS	GET	200	application/x-javascript	4.66 KB	313.5 ms	script X-P2P-PeerDistEx: MinContention	nformation=1.0, MaxCont.
https://i1.services.social.microsoft.com/search/Widgets/ jquery-2.1.0.min.js	HTTPS	GET	0K 304	application/x-javascript	(from cache)	10.86 ms	Response Headers script Cache-Control: private	
https://ajax.aspnetcdn.com/ajax/jQuery/ Combined.css?resources=0:Home.1,0:HeroRotator.1,2:jqu	HTTPS	GET	Not Modified 200	text/css	2.71 KB			
https://www.visualstudio.com/ analytics.js	HTTPS	GET	OK 304	text/javascript	(from ca		нттрс	_
https://www.google-analytics.com/ Bootstrap.is	HTTPS	GET	304	application/x-iavascript	(from		11113	
https://nexus.ensighten.com/msvscs/	HTTPS	GET	Not Modified	text/impercient				
https://ots.optimize.webtrends.com/ots/api/js-4.1/204335/WT3	innes	GLI	OK	textojavascript		X		
serverComponent.php?r=578293.7306700915&ClientID= https://nexus.ensighten.com/msvscs/prod/	HTTPS	GET	200 OK	text/javascript		^.	ASIN.1	
jquery.min.js https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/	HTTPS	GET	304	text/javascript	(from	Т		
321c0db7485fb02e24b7b5ddedd3dbd8,js?conditionId0= https://nexus.ensighten.com/msvscs/prod/code/	HTTPS	GET	304 Not Modified	application/x-javascript	(from		L TL	S
platform.js https://www.microsoft.com/content/f/feeds/msdn/en-us/	HTTPS	GET	304 Not Modified	application/x-javascript	(from	<u>ا</u> .		
ai.0.js https://az416426.vo.msecnd.net/scripts/a/	HTTPS	GET	304 Not Modified	application/x-javascript	(from	- 1	***	
a.js;m=11087202615936;cache=0.2511960009749005?	HTTP/2	GET	200	text/javascript	739 B	•		
2 https://nts.ontimize.webtrends.com/nts/ani/is_41/204335/041H	HTTPS	GET	200 OK	text/javascript		RS	A SHA 🖛	
0f90383d2deb0c0878e399d284d548ae.js?conditionId0=2	HTTPS	GET	200	application/x-javascript	(from	- E(
26f2e6c1568be56d0b08f2295feb40c3.js?conditionId0=28	HTTPS	GET	200	application/x-javascript	(from			
rio.ashx?ootc=200647323	HTTPS	GET	200	text/javascript	4.38 K	Cryp	to Algorithms	
ms.js	HTTPS	GET	200	application/x-javascript	(from			•
ncups//cmicrosoft.com/ sizzle.min.map https://cwebtrends.com/acs/common/js/custom/sizzle/	HTTPS	GET	200 OK	text/plain	27.38 KB		Network buffe	rs

Long-term identities: X.509

Public-Key Infrastructure (Certificate Chains)

Designed in 1984; widely criticized but hard to replace HTTPS is just one application

Same complexity as TLS?

ASN.1 grammar; many extensions and interpretations 50% of "TLS attacks" are in fact X.509 attacks

Recent initiatives

Global scans for millions of certificates Certificate pinning & transparency Let's encrypt! <u>https://letsencrypt.org/</u>

Verification?

Complex ambiguous format Certificate issuance and revocation policies



Cryptographic Algorithms for HTTPS

Algorithms get broken & replaced over time

Security relies on probabilistic cryptographic assumptions (who knows?) Modern design & implementations select between various algorithms & implementations for the same core functionality

~30 standard algorithms

- Hash and key-derivation functions (SHA256)
- Symmetric cryptography (AES_GCM, AES_CBC)
- Public-key encryption and signing
- Elliptic curves (NIST, 25519, 4Q)

High-performance

AES_GCM takes 0.46 cycle/byte on Intel Skylake Hand-tuned, low-level, architecture-specific





Microsoft Research - Inria JOINT CENTRE



Verified Secure Implementations for the HTTPS Ecosystem miTLS & Everest*