Provable security of Internet cryptography protocols

Douglas Stebila

McMaster University

UNIVERSITY OF WATERLOO

Based on joint works with Florian Bergsma, Katriel Cohn-Gordon, Cas Cremers, Ben Dowling, Marc Fischlin, Felix Günther, Luke Garratt, Florian Kohlar, Jörg Schwenk

Funding acknowledgements:
ATN-DAAD, ARC

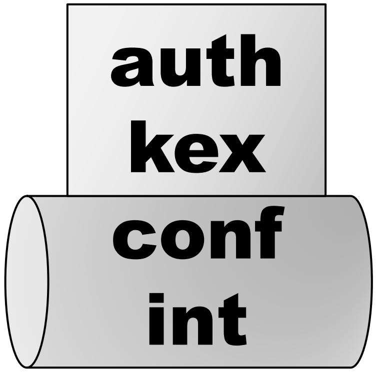
NSERC CRSNG

Summer School on real-world crypto and privacy • Šibenik, Croatia • June 15, 2018
https://www.douglas.stebila.ca/research/presentations

Introduction

Establishing secure channels

- Primary goal of much of cryptography: enabling secure communication between two parties



Authenticated key exchange

auth kex

- **Goal: two parties establish a random shared session key between them; the key is unknown to any active adversary**

- Variety of very complex security models which capture subtly different properties
  - BR93
  - BR95
  - BJM97
  - BPR00
  - CK01
  - CK02
  - LLM07 (eCK)
  - …

AKE setting

auth kex

- Multiple parties, each with a long-term secret key / public key pair
- Distribution of public keys is typically outside the scope of the protocol (e.g., assume a PKI or magical key delivery fairy*)
- A "session" is an instance of the protocol run at a party
- Each party can run multiple sessions in parallel or sequentially
- Each session eventually "accepts" (outputting a session key and name of a peer), or "rejects"

* Some attempts to model PKI in AKE: e.g. [Boyd et al, ESORICS 2013]

AKE security goals

auth
kex

- **Session key indistinguishability:**
  - Two parties establish a session key that is indistinguishable from random
- **Server-to-client authentication:**
  - If a client accepts in a session, then there exists a (unique) "matching" session at the peer
    - A party should accept only if its peer really was active in this sequence of communications
- **Client-to-server authentication**

AKE attack powers, informally

auth kex

- Adversary can **control all network communications**, including:
  - Directing parties to send protocol messages
  - Changing the destination of a protocol message
  - Reordering, dropping, changing a protocol message
  - Creating protocol messages
- Adversary can **reveal certain secret values** held by parties

AKE attack powers, formally

auth kex

Adversary can access several oracles:

Some oracles simulate "normal" operation of the protocol:

- **Send**(U, i, m): Send message m to instance i of user U

AKE attack powers, formally

auth kex

Adversary can access several oracles:

Some oracles enable the experiment to be executed:

- **Test**(U, i): A hidden bit b is chosen. If b=0, the adversary is given the real session key for user U's i'th session; if b=1, the adversary is given a uniform random string of the same length. The adversary must output a guess of b at the end of its execution.

AKE attack powers, formally

auth kex

Adversary can access several oracles:

Some oracles allow the attacker to learn certain secret values:

- **RevealLongTermKey**(U): Returns party U's long-term secret key
- **RevealRandomness**(U, i): Returns any randomness used by party U in session i
- **RevealSessionState**(U, i): Returns party U's local state in session i
- **RevealSessionKey**(U, i): Returns the session key derived by party U in session i

AKE freshness

auth kex

- Since some oracles allow the adversary to learn secret values, we have to prohibit the adversary from learning so many values that it could trivially compute the test session's session key: "**freshness**"
- Different combinations of prohibited queries lead to different security properties and different AKE security models in the literature
  - E.g. eCK versus CK
- Also introduces a notion of "matching" or "partnering"

Authenticated encryption

- **Goal: two parties can transmit messages in a confidential way and be sure they are not interfered with (integrity)**
- Symmetric authenticated encryption assumes parties have a uniformly random shared secret key to begin with
- Variety of increasingly complex security definitions to capture increasingly realistic security properties:
  - [Bellare, Namprempre ASIACRYPT 2000]
  - [Rogaway CCS 2002] – with associated data
  - [Bellare, Kohno, Namprempre; CCS 2002]
  - [Kohno, Palacio, Black eprint 2003/177]
  - [Paterson, Ristenpart,, Shrimpton ASIACRYPT 2011]
  - [Boldyreva, Degabriele, Paterson, Stam EUROCRYPT 2012]
  - [Fischlin, Günther, Marson, Paterson CRYPTO 2015]
  - [Shrimpton, yesterday's talk]
  - …

AE models

Realism

- Partially-specified authenticated encryption for streams
- Authenticated encryption for streams
- Buffered stateful authenticated encryption
- Stateful length-hiding authenticated encryption
- Stateful authenticated encryption
- Authenticated encryption
- Confidentiality
- Integrity

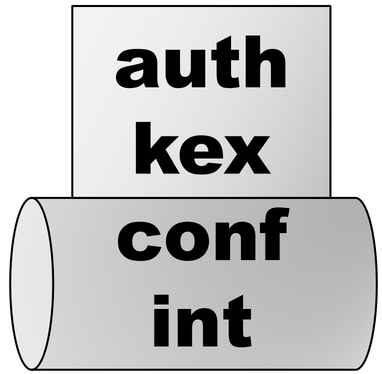
Complexity

Inspired by Tom Shrimpton's talk yesterday

Stateful length-hiding authenticated encryption with associated data

- "Authenticated": integrity of ciphertexts
- "Encryption": confidentiality of plaintexts
- "Associated data": integrity of some associated "header" data which is not necessarily confidential (maybe not even transmitted)
- "Stateful": cryptographic protection against reordering of ciphertexts
- "Length-hiding": adversary can't distinguish between short and long messages (up to a maximum length)

Composing AKE and AE



To establish a secure channel:

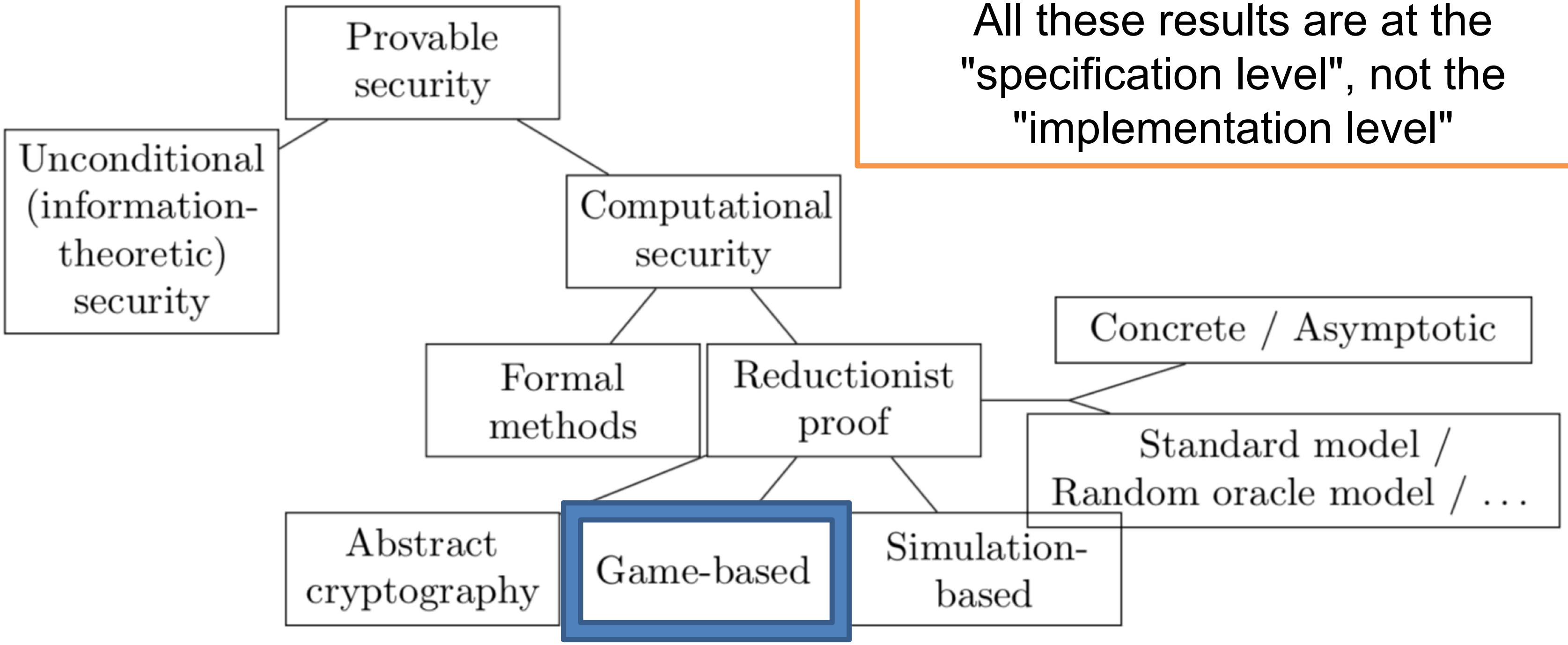
1. Use an AKE protocol to establish a shared secret key
2. Use the shared secret key in an authenticated encryption scheme
3. Apply a composability result, e.g. [Canetti, Krawczyk EUROCRYPT 2001]

We're done!

"Provable security"

Be aware of limitations of provable security methodology, e.g. Koblitz and Menezes

All these results are at the "specification level", not the "implementation level"



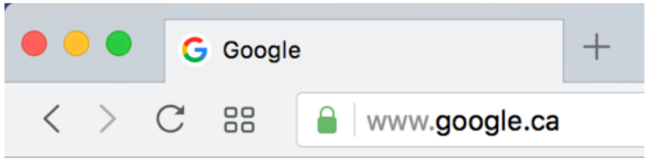
TLS 1.2

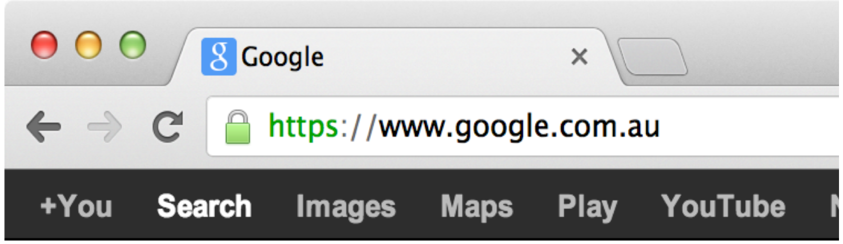
History of TLS

- SSL: Secure Sockets Layer
- Proposed by Netscape
  - SSLv2: 1995
  - SSLv3: 1996

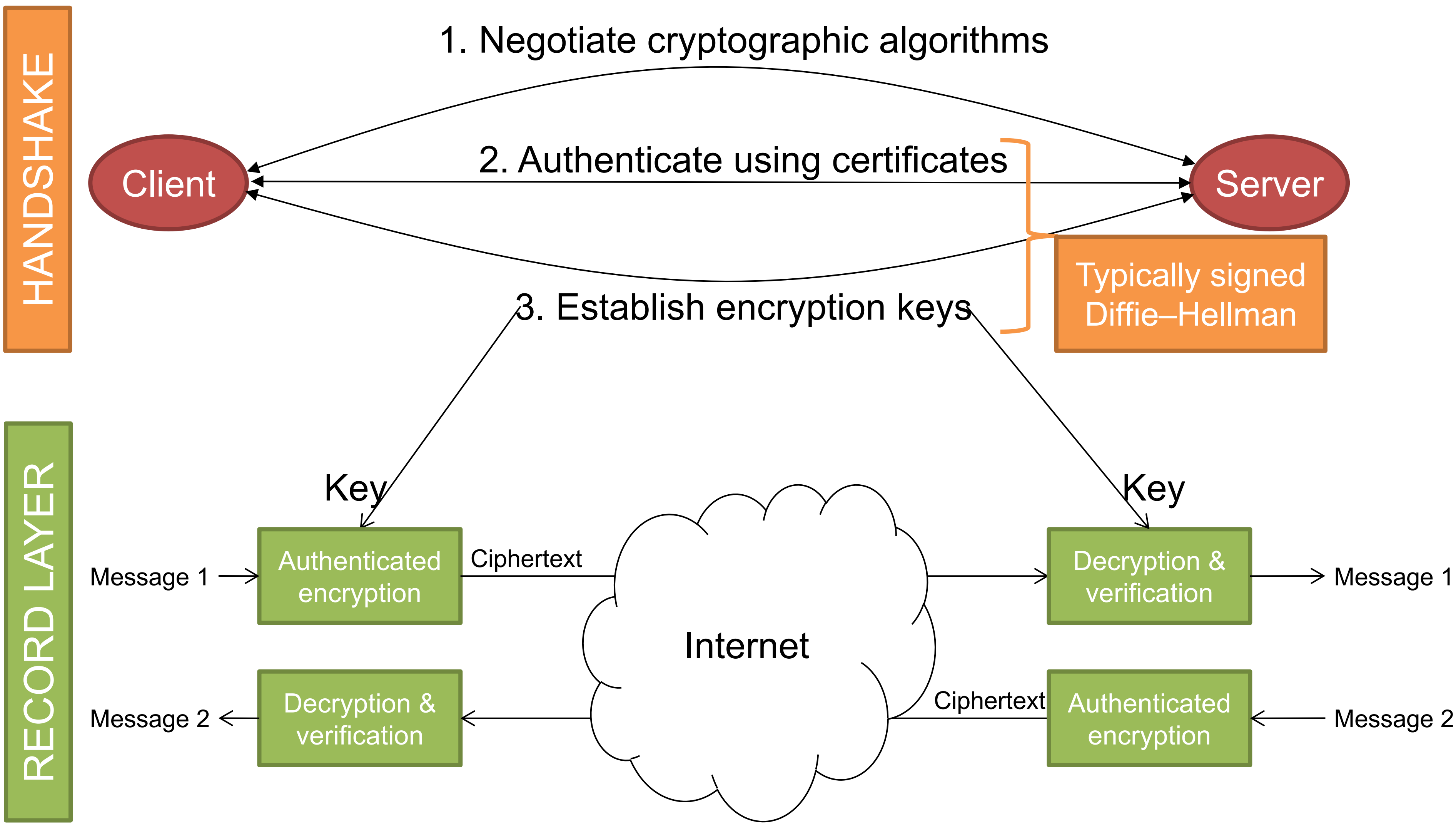
- TLS: Transport Layer Security
- IETF Standardization of SSL
  - TLSv1.0 = SSLv3: 1999
  - TLSv1.1: 2006
  - TLSv1.2: 2008
  - TLSv1.3: 2018?

- HTTPS: HTTP (Hypertext Transport Protocol) over SSL

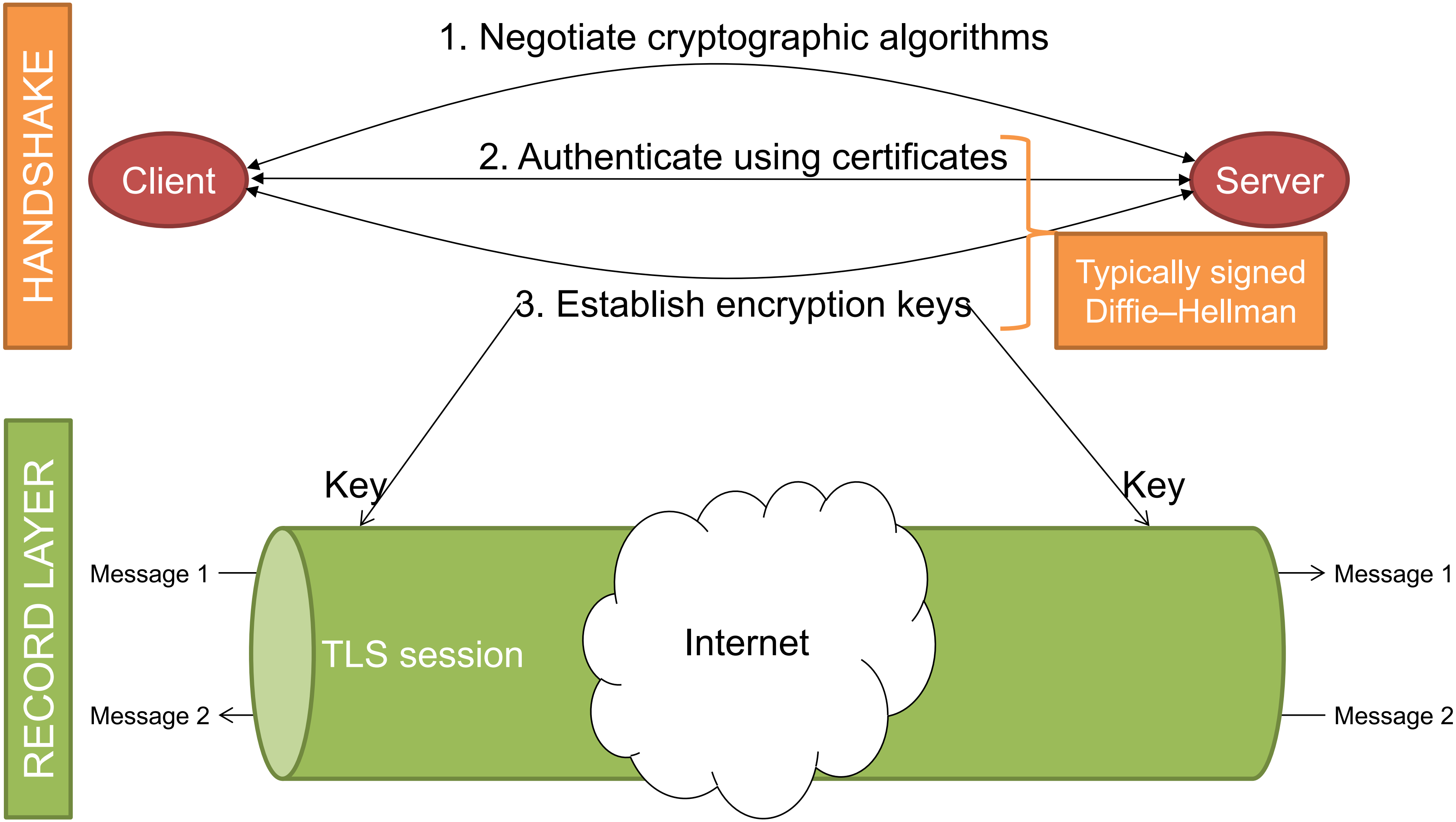




SSL/TLS Protocol



SSL/TLS Protocol



Structure of TLS 1.2

HANDSHAKE PROTOCOL

Negotiation of cryptographic parameters

Authentication (one-way or mutual) using public key certificates

Establishment of a master secret key

Derivation of encryption and authentication keys

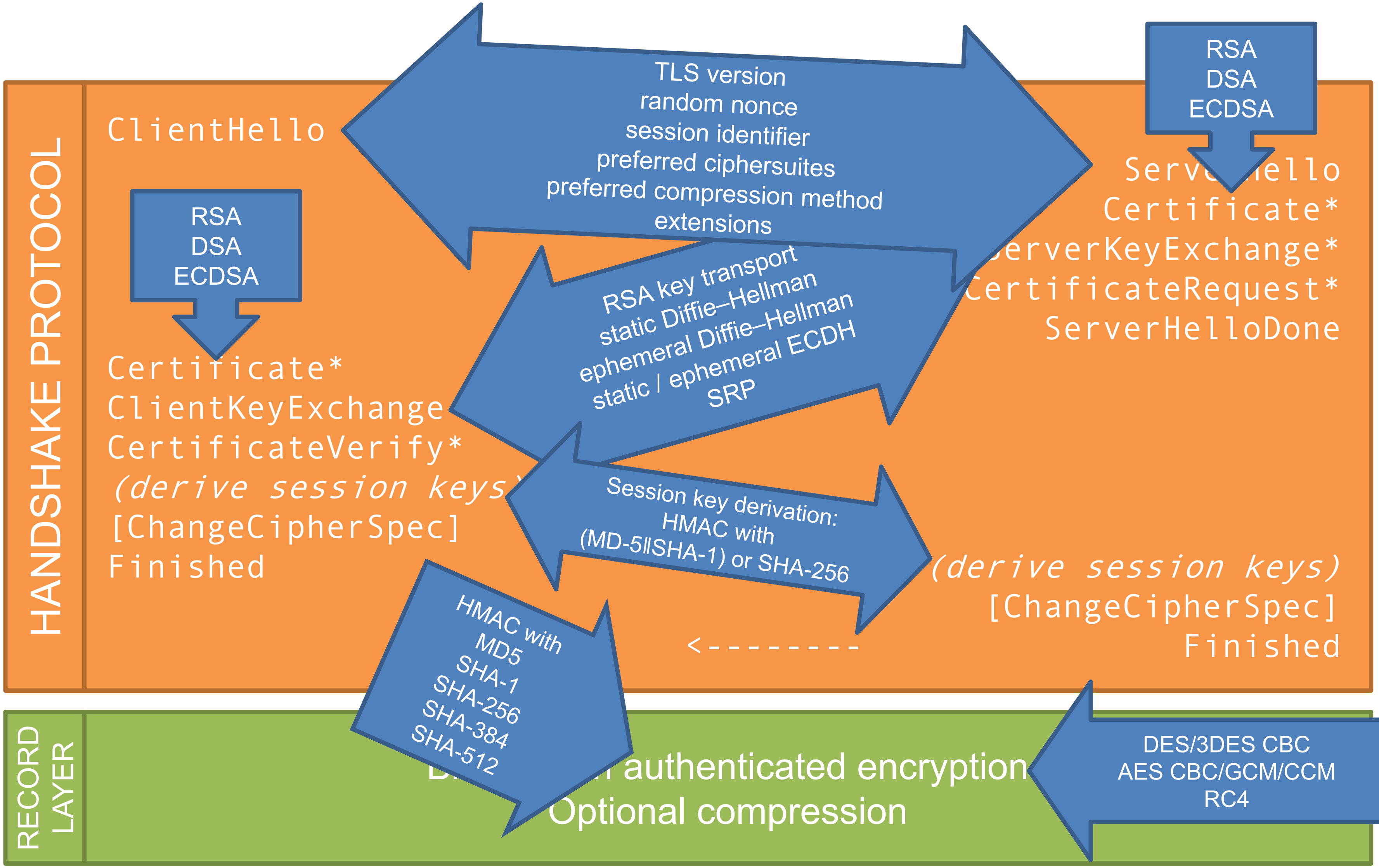
Key confirmation

ALERT PROTOCOL

RECORD LAYER

Bi-direction authenticated encryption
Optional compression

Structure of TLS ≤1.2



Challenges with proving TLS ≤1.2 secure

ChangeCipherSpec: "I will encrypt all subsequent messages"

Finished: MAC(session key, handshake transcript)

HANDSHA

```
(derive session keys)
[ChangeCipherSpec]
Finished
                    -------->
                                         (derive session keys)
                                            [ChangeCipherSpec]
                    <--------                         Finished
```

Note that Finished message is encrypted due to ChangeCipherSpec

Challenges with proving TLS ≤1.2 secure

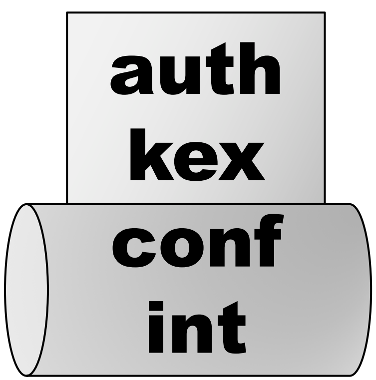
- Recall AKE security goal: session key indistinguishability:
  - Adversary is given either the real session key or a random session key, asked to decide which

- In TLS ≤1.2, adversary is given ciphertexts (encryptions & MACs) of known plaintexts under the real session key
- To trivially distinguish real from random, trial decrypt the Finished message and see if it is valid
- **Conclusion: TLS ≤1.2 handshake is not a secure AKE protocol**

Is TLS secure?

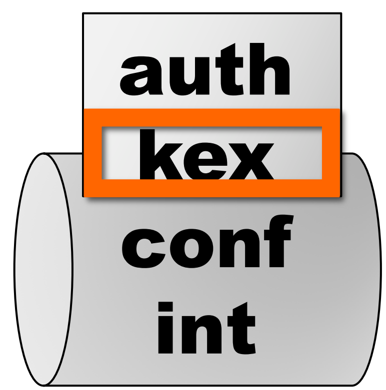
Ideal

- Prove TLS handshake is a secure AKE
- Prove TLS record layer is a secure AE
- Apply composability result



Problem

- TLS handshake sends messages encrypted under the session key
- TLS handshake is not a secure AKE
- Can't apply composability



Early works on proving SSL/TLS secure

1996 SSL v3.0 standardized

2001 Some variant of one ciphersuite of the TLS record layer is a secure encryption scheme [Kra01]

2002 Truncated TLS handshake using RSA key transport is a secure authenticated key exchange protocol [JK02]

2008 Truncated TLS handshake using RSA key transport or signed Diffie–Hellman is a secure AKE [MSW08]

"some variant"… "truncated TLS"… limited ciphersuites

Rule #1 for making cryptographers' lives hard

Use the session key during the protocol so the AKE can't be composed with the AE

See TLS ≤1.2, SSH, EMV, …

Progress in proving TLS 1.2

Progress in proving TLS 1.2

1996
SSL v3.0 standardized

2011
Some modes of TLS record layer are secure authenticated encryption schemes

[PRS11]

2012
Unaltered full signed Diffie–Hellman ciphersuite is a secure channel

[JKSS12]

2013
Most unaltered full TLS ciphersuites are a secure channel

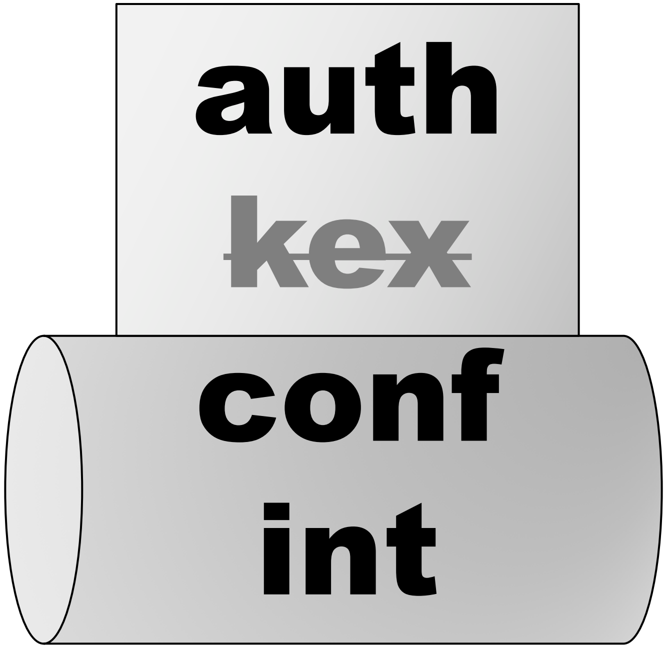
[KSS13, KPW13, BFKPS13]

"unaltered"… "full"… "most ciphersuites"

Authenticated and Confidential Channel Establishment (ACCE)

- Captures:
  - entity authentication
  - confidentiality and integrity of messages
- In a single "monolithic" security definition

- Avoids composability problems by directly proving the full "secure channel" property



[Jager, Kohlar, Schage, Schwenk CRYPTO 2012]

Security models

AKE + AE

- AKE normal operations:
  - Send
- AKE learn some secrets:
  - RevealLongTermKey
  - RevealSessionKey
- AKE experiment:
  - Test
- AKE Goal: Guess real/random hidden bit

- AE normal operations:
  - Enc
  - Dec
- AE Goal: Distinguish messages or forge ciphertext

ACCE

- Normal operations:
  - Send
- Learn some secrets:
  - RevealLongTermKey
  - RevealSesionKey
- Experiment:
  - Encrypt
  - Decrypt

Like Enc/Dec for stateful length-hiding authenticated encryption, per TLS session

- ACCE goal: distinguish messages or forge ciphertext

On the Security of TLS-DHE in the Standard Model[1]

Tibor Jager
Horst Görtz Institute for IT Security
Bochum, Germany
tibor.jager@rub.de

Florian Kohlar
Horst Görtz Institute for IT Security
Bochum, Germany
florian.kohlar@rub.de

Sven Schäge[2]
University College London
United Kingdom
s.schage@ucl.ac.uk

Jörg Schwenk
Horst Görtz Institute for IT Security
Bochum, Germany
joerg.schwenk@rub.de

February 20, 2013

Theorem: Signed Diffie–Hellman with a suitable record layer mode is a secure ACCE protocol, under suitable assumptions on the underlying cryptographic building blocks.

We're done!

Provable security of TLS

Crypto primitives
- RSA, DSA, ECDSA
- Diffie–Hellman, ECDH
- HMAC
- MD5, SHA1, SHA-2
- DES, 3DES, RC4, AES
-

Ciphersuite details
- Data structures
- Key derivation
- Encryption modes, IVs
- Padding

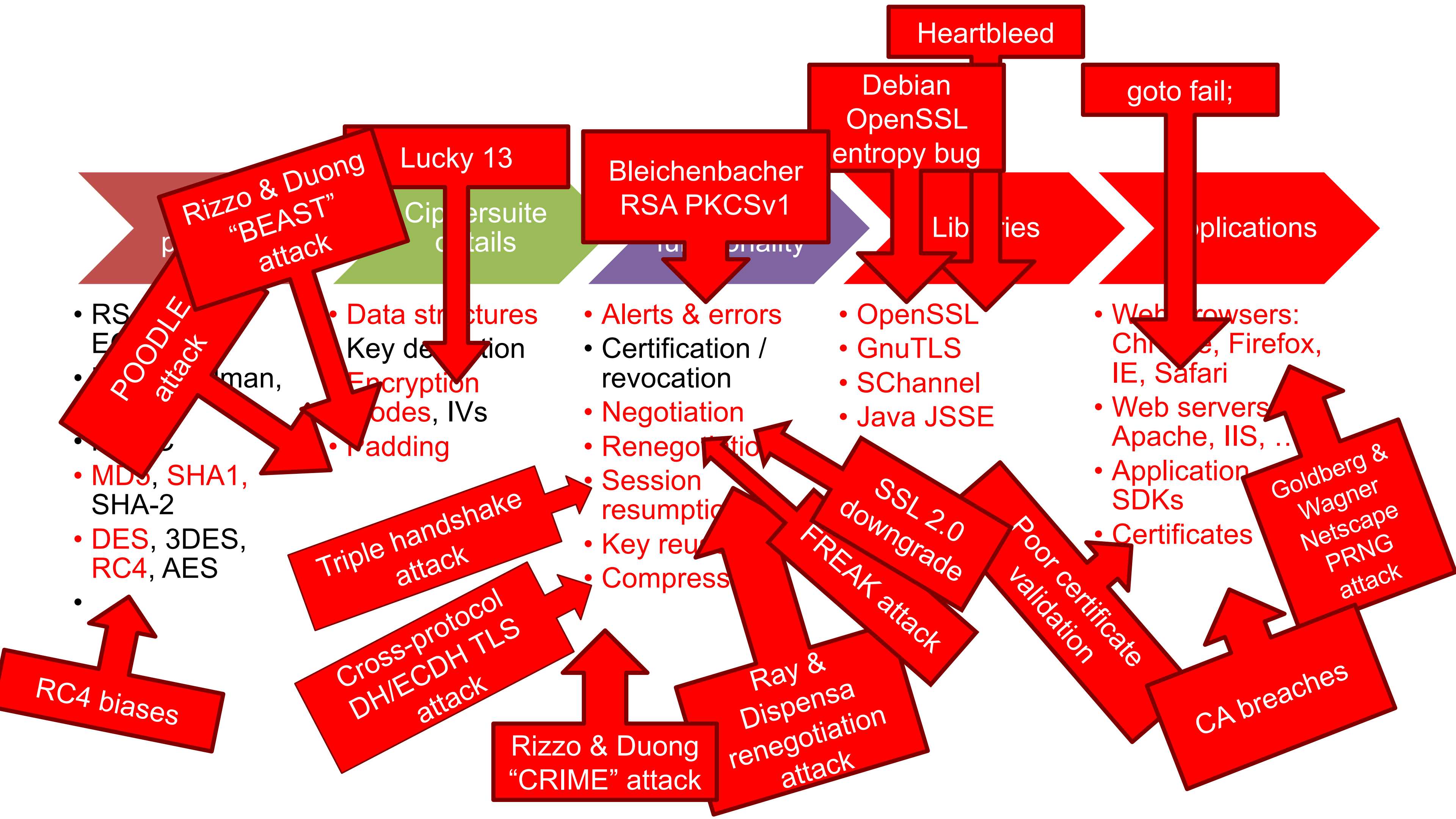
Advanced functionality
- Alerts & errors
- Certification / revocation
- Negotiation
- Renegotiation
- Session resumption
- Key reuse
- Compression

Libraries
- OpenSSL
- GnuTLS
- SChannel
- Java JSSE

Applications
- Web browsers: Chrome, Firefox, IE, Safari
- Web servers: Apache, IIS, …
- Application SDKs
- Certificates

Provably secure "cryptographic core"
sLHAE: TLS AES-GCM
ACCE results: TLS-DHE, -RSA, -DH, -PSK

Real-world attacks on TLS



(Selected) advanced security properties of TLS 1.2

Negotiation

Renegotiation

Resumption

Negotiation

- TLS isn't a fixed combination of cryptographic algorithms
- Parties **negotiate** which combinations of algorithms
  - Based on their own preferences and support
  - In TLS ≤1.2, simultaneous negotiation the full combination of algorithms, called a "ciphersuite"
- Parties also negotiate other aspects, such as what **version** of TLS to use

Negotiation and downgrading

- Some ciphersuites and versions may be weaker than others but still be supported for backwards compatibility with old implementations

- Most clients will do the "version downgrade dance" to attempt to find a mutually compatible configuration

TLS 1.2
- Client attempts handshake
- Version failure response [unauthenticated]

TLS 1.1
- Client attempts handshake
- Version failure response [unauthenticated]

TLS 1.0
- Client attempts handshake
- Version failure response [unauthenticated]

SSL 3.0
- Client attempts handshake
- Success! [but not really]

Version downgrade attacks

- POODLE attack [Möller, Duong, Kotowicz 2014]
  - Utilitizes downgrade to SSL 3
- Countermeasure: Version Fallback Signalling Ciphersuite Value
  - TLS extension/hack to detect version downgrade attacks
  - Have to be clever to ensure backwards compatibility across the TLS ecosystem

Ciphersuite downgrade attacks

- Client and server both support both good_ciphersuite and weak_ciphersuite, would prefer to agree on good_ciphersuite
- FREAK attack [Beurdouche et al. SP 2015]
- Logjam attack [Adrian et al. CCS 2015]

- Real ClientHello: good_, weak_
- Adversary: send fake ClientHello with only weak_
- ServerHello: respond with weak_
- Adversary: relay rest of handshake
- Adversary: must forge MAC in Finished message to make parties agree on mismatching transcript, but may be possible due to weak_ciphersuite

Modelling negotiation

- A full analysis of TLS would model it as a suite of protocols with different versions and ciphersuites
- Security goal would include ability to cause parties to negotiate at a mutually-sub-optimal configuration
- But these different versions/ciphersuites often share long-term keys making composition tricky

- Theorem: TLS with version negotiation using the downgrade dance and the "version fallback signaling ciphersuite value" countermeasure is as secure as the ACCE authentication security of the weakest TLS version.

[Dowling, Stebila ACISP 2015]

Renegotiation

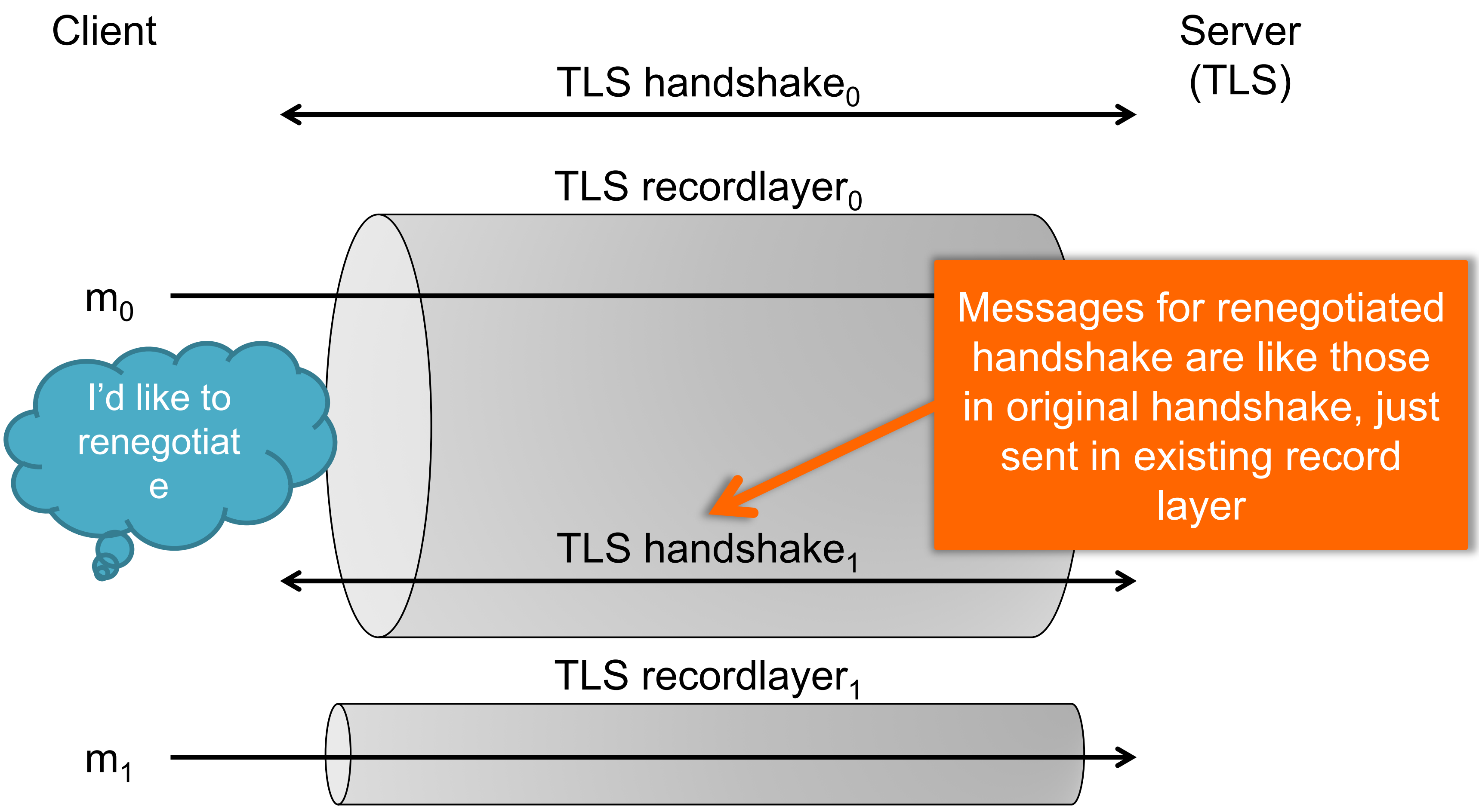
Renegotiation allows parties in an established TLS channel to create a new TLS channel that continues from the existing one.

Once you've established a TLS channel, why would you ever want to renegotiate it?

- Change cryptographic parameters
- Change authentication credentials
- Identity hiding for client
  - second handshake messages sent encrypted under first record layer
- Refresh encryption keys
  - more forward secrecy
  - record layer has maximum number of encryptions per session key

Renegotiation in TLS ≤1.2

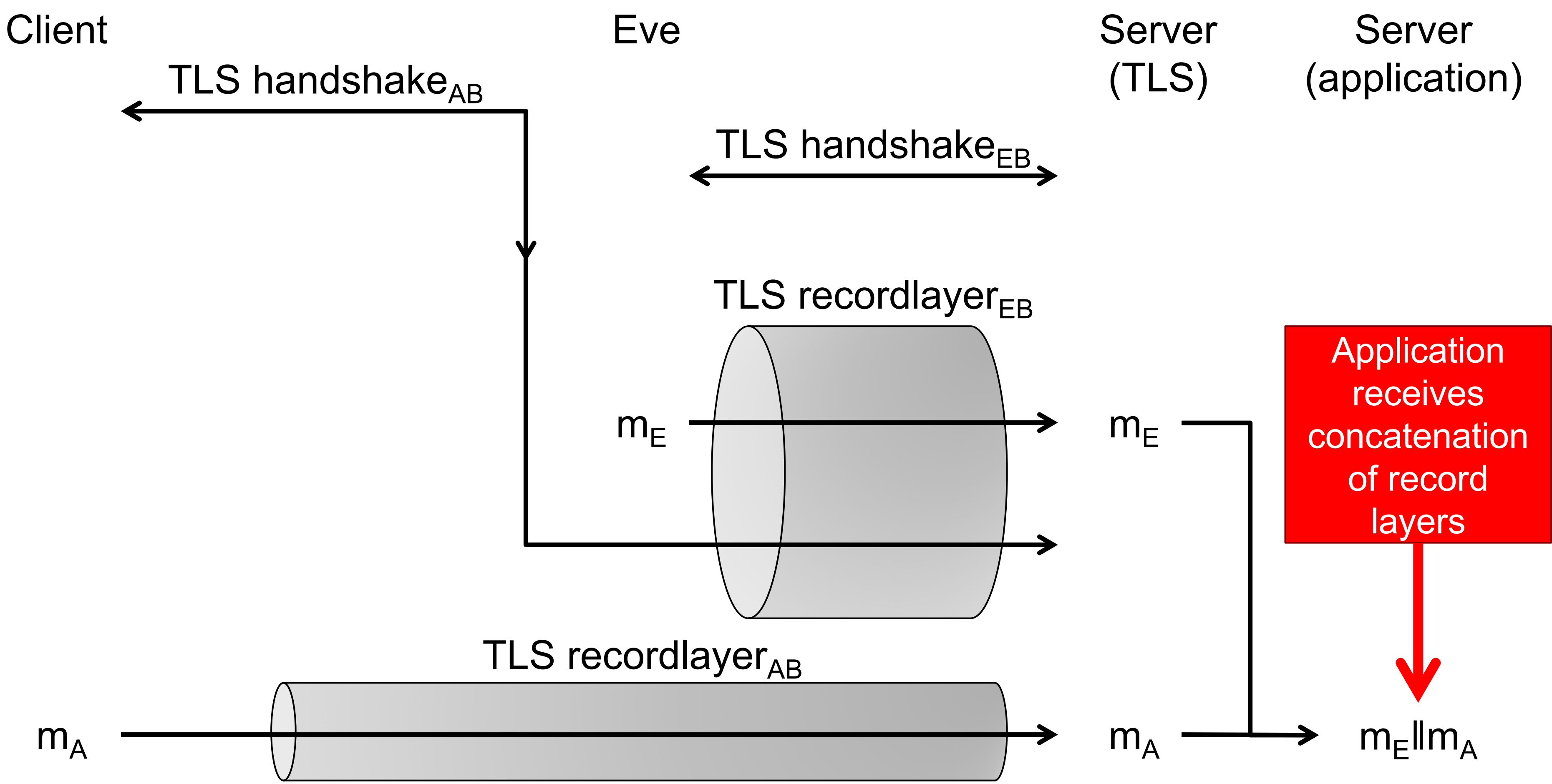
(pre-November 2009)



TLS Renegotiation “Attack”

Ray & Dispensa, November 2009

Not an attack on TLS, but on how applications misuse TLS



Modelling renegotiation security

Q: What property should a secure renegotiable protocol have?

A: Whenever two parties successfully renegotiate, they are assured they have the exact same view of everything that happened previously.

- Every time we accept, we have a matching conversation of previous handshakes and record layers.

Weakly secure renegotiable ACCE

Definition

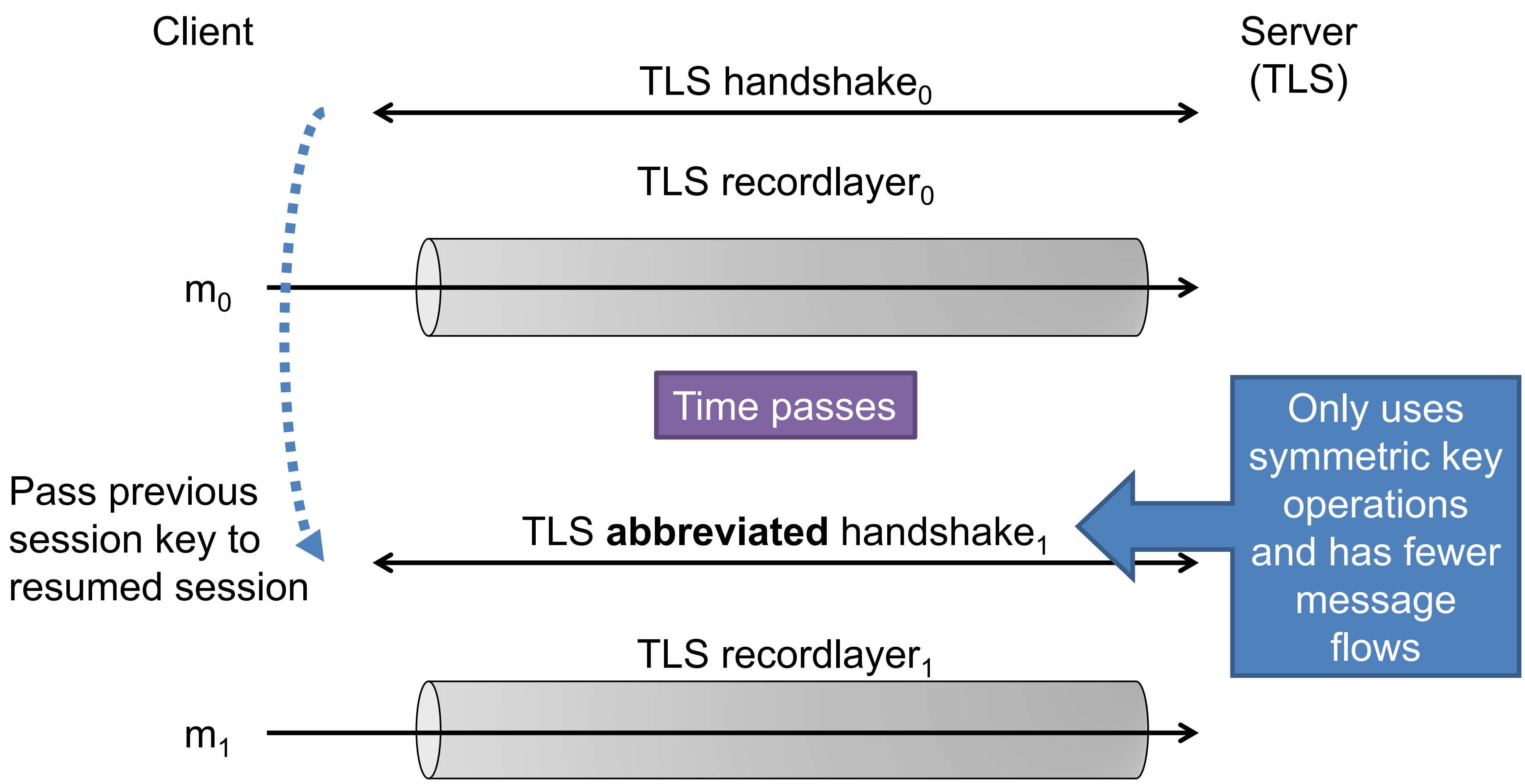
When a party successfully renegotiate a new phase, its partner has a phase with a matching handshake and record layer transcript, *provided no previous phase's session key was revealed.*

TLS

- TLS without fixes is not a weakly secure renegotiable ACCE.
- TLS with RFC 5746 fixes is a weakly secure renegotiable ACCE.
  - (This is probably good enough.)

[Giesen, Kohlar, Stebila, CCS 2013]

Session resumption



Triple handshake attack

- Man-in-the-middle attack on three consecutive handshakes
- Relies on session resumption and renegotiation
  - works even with countermeasures against renegotiation attack
- Works due to lack of binding between sessions during session resumption

[Bhargavan et al. S&P 2014]

Summarizing attacks on TLS≤1.2

Core cryptography		
RSA PKCS#1v1.5 decryption	Side channel – Bleichenbacher	1998*, 2014
DES	Weakness – brute force	1998
MD5	Weakness – collisions	2005
RC4	Weakness – biases	2000*, 2013
RSA export keys	FREAK	2015
DH export keys	Logjam	2015
RSA-MD5 signatures	SLOTH	2016
Triple-DES	Sweet32	2011*, 2016
Crypto usage in ciphersuites		
CBC mode encryption	BEAST	2002*, 2011
Diffie–Hellman parameters	Cross-protocol attack	1996*, 2012
MAC-encode-encrypt padding	Lucky 13	2013
CBC mode encryption + padding	POODLE	2014
TLS protocol functionality		
Support for old versions	Jager et al., DROWN	2015, 2016
Negotiation	Downgrade to weak crypto	1996, 2015
Termination	Truncation attack	2007, 2013
Renegotiation	Renegotiation attack	2009
Compression	CRIME, BREACH, HEIST	2002*, 2012,16
Session resumption	Triple-handshake attack	2014

Summarizing attacks on TLS≤1.2

Implementation – libraries		
OpenSSL – RSA	Side-channel	2005, 2007
Debian OpenSSL	Weak RNG	2008
OpenSSL – elliptic curve	Side-channel	2011–14
Apple – certificate validation	goto fail;	2014
OpenSSL – Heartbeat extension	Heartbleed	2014
Multiple – certificate validation	Frankencerts	2014
NSS – RSA PKCS#1v1.5 signatures	BERserk (Bleichenbacher)	2006*, 2014
Multiple – state machine	CCS injection, SMACK	2014, 2015
Implementation – HTTP-based applications		
Netscape	Weak RNG	1996
Multiple – certificate validation	"The most dangerous code..."	2012
Application-level protocols		
HTTP	SSL stripping	2009
HTTP server virtual hosts	Virtual host confusion	2014
IMAP/POP/FTP	STARTTLS command injection	2011

TLS 1.3

TLS 1.3

In 2014, IETF started to develop a new version of TLS, now called TLS 1.3

Goals:

1. Deprecate old cryptography, use modern crypto
2. Encrypt parts of the handshake
3. Reduce latency of handshake establishment by providing modes with fewer roundtrips ("0-RTT")
4. General changes to improve/simplify protocol logic

Structure of TLS 1.3 full handshake

HANDSHAKE PROTOCOL

```
ClientHello
+ key_share, ...
                          -------->
                                                   ServerHello
                                                + key_share, ...
                                 (derive handshake traffic key)
                                        [Encrypted Extensions]
                                        [CertificateRequest*]
                                                [Certificate*]
                                          [CertificateVerify*]
                                                    [Finished]
                          <--------
                               (derive application traffic key)
                                          [[application data]]
(derive handshake traffic key)
[Certificate*]
[CertificateVerify*]
[Finished]
                          -------->
(derive application traffic key)
```

Client sends ephemeral public key right away

Server encrypts parts of handshake under temporary key

Server sends application data under a new key

RECORD LAYER

Bi-direction authenticated encryption

Structure of TLS 1.3 short handshake in 0-RTT mode

HANDSHAKE PROTOCOL

```
ClientHello
+ key_share, pre_shared_key, ...
(derive early application traffic key)
[[[application data]]]
                                    -------->
                                                    ServerHello
                                                + key_share, ...
                                  (derive handshake traffic key)
                                          [Encrypted Extensions]
                                    <--------         [Finished]
                                (derive application traffic key)
                                            [[application data]]
(derive handshake traffic key)
[Finished]                          -------->
(derive application traffic key)
```

Client includes reference to a pre-shared symmetric key (probably exported from a previous session)

Client sends application data encrypted under temporary key

RECORD LAYER

Bi-direction authenticated encryption

Challenges with proving TLS 1.3 secure

HANDSHAKE PROTOCOL

ClientHello
+ key_share, ...

There are two (or three) (or more) keys per session.

Some of the keys are used for encrypting data while the rest of the handshake session continues.

ServerHello
key_share, ...
e traffic key)
ted Extensions]
ficateRequest*]
[Certificate*]
ificateVerify*]
[Finished]
n traffic key)

[[application data]]

(derive handshake traffic key)
[Certificate*]
[CertificateVerify*]
[Finished] -------->
(derive application traffic key)

RECORD LAYER

Bi-direction authenticated encryption

Multi-stage AKE

- Originally introduced by [Fischlin, Günther CCS 2014] for analyzing Google's QUIC protocol

- Each session can derive multiple session keys for each **stage**

- Each stage's session key should be indistinguishable from random
- Even under certain secrets being revealed
  - Long-term secret keys
  - Session keys of other sessions and other stages of same session
  - Don't consider leakage of randomness/internal state as not part of TLS 1.3 design goals

- Also need a model that works when keys are used to encrypt data while handshake continues

Selected provable security results on TLS 1.3 handshakes

[Dowling, Fischlin, Günther, Stebila CCS 2015/TRON 2016/thesis]

- TLS 1.3 draft-10&16 full ECDHE handshake establishes
  - random-looking session keys for every stage
  - forward secrecy for all of these
  - anonymous/unilateral/mutual authentication
  - key independence (leakage of key in one stage does not affect another stage)
- under suitable assumptions.

- Similarly for short handshake, without consideration of 0-RTT application data.

- Suitable for modular composition with authenticated encryption modelling of record layer

[Fischlin, Günther EuroS&P 2017]

- TLS 1.3 draft-14 short handshake in 0-RTT mode establishes
  - random-looking session keys for every stage
  - NO forward secrecy for 0-RTT keys
  - NO replay protection for 0-RTT keys and data

[Krawczyk, Wee EuroS&P 2016] [Li, Zhang, Feng, Mu S&P 2016]

SSH

Is SSH secure?

2006
SSH v2 standardized

2004
Some variant of SSH encryption is secure
[BKN04]

2009-10
Attack on SSH encryption, fixed version is secure
[APW09, PW10]

2011
Truncated SSH handshake using signed Diffie–Hellman is a secure AKE
[Wil11]

“some variant”… “truncated SSH”

SSH using Signed-DH is ACCE-secure

Theorem: Assuming

- the signature scheme is secure,
- the computational Diffie–Hellman problem is hard,
- the hash function is random,
- and the encryption scheme is a secure buffered stateful authenticated encryption scheme,

then an individual signed-Diffie–Hellman SSH ciphersuite is ACCE-secure.

[Bergsma, Dowling, Kohlar, Schwenk, Stebila, ACM CCS 2014]

Cryptographic algorithms in SSH

- Authentication:
  - RSA signatures
  - DSA-SHA1
  - ECDSA-SHA2
  - X509-RSA signatures
  - X509-DSA-SHA1
  - X509-ECDSA-SHA2
- Key exchange:
  - DH explicit group SHA1
  - DH explicit group SHA2
  - DH group 1 SHA1
  - DH group 14 SHA1
  - ECDH-nistp256-SHA2
  - ECDH-nistp384-SHA2
  - ECDH-nistp521-SHA2
  - ECDH-*-SHA2
  - GSS-group1-SHA1-*
  - GSS-group14-SHA1-*
  - GSS explicit group SHA1
  - RSA1024-SHA1
  - RSA2048-SHA2
  - ECMQV-*-SHA2
- Encryption:
  - 3des-cbc
  - blowfish-cbc
  - twofish256-cbc
  - twofish-cbc
  - twofish192-cbc
  - twofish128-cbc
  - aes256-cbc
  - aes192-cbc
  - aes128-cbc
  - serpent256-cbc
  - serpent192-cbc
  - serpent128-cbc
  - arcfour
  - idea-cbc
  - cast128-cbc
  - des-cbc
  - arcfour128
  - arcfour256
  - aes128-ctr
  - aes192-ctr
  - aes256-ctr
  - 3des-ctr
  - blowfish-ctr
  - twofish128-ctr
  - twofish192-ctr
  - twofish256-ctr
  - serpent128-ctr
  - serpent192-ctr
  - serpent256-ctr
  - idea-ctr
  - cast128-ctr
  - AEAD_AES_128_GCM
  - AEAD_AES_256_GCM
- MACs:
  - hmac-sha1
  - hmac-sha1-96
  - hmac-md5
  - hmac-md5-96
  - AEAD_AES_128_GCM
  - AEAD_AES_256_GCM
  - hmac-sha2-256
  - hmac-sha2-512

Theorem implies each of these are secure

RSA signatures

- Diffie–Hellman group 14
- AES-128
- HMAC-SHA-1

ECDSA signatures

- Elliptic curve Diffie–Hellman nistp256
- AES-128
- HMAC-SHA-256

What if I use the same signature key with different key exchange algorithms?

14

ECI
signa

In practice, TLS and SSH servers use the same long-term key for all ciphersuites

llman

- AES-128
- HMAC-SHA-256

Long-term key reuse across ciphersuites

Is this secure?

Even if a ciphersuite is provably secure on its own, it may not be secure if the long-term key is shared between two ciphersuites.

TLS
Individual ciphersuites are secure without key reuse.

[MVVP12] attack => insecure with key reuse.

SSH
Individual ciphersuites are secure without key reuse.

???

Key re-use framework and security proof

New security definition for key reuse

Abstract framework for proving safe key reuse

careful: needs to work for SSH but not TLS

Security proof of multiple full SSH suites used simultaneously

Conclusions

AE models

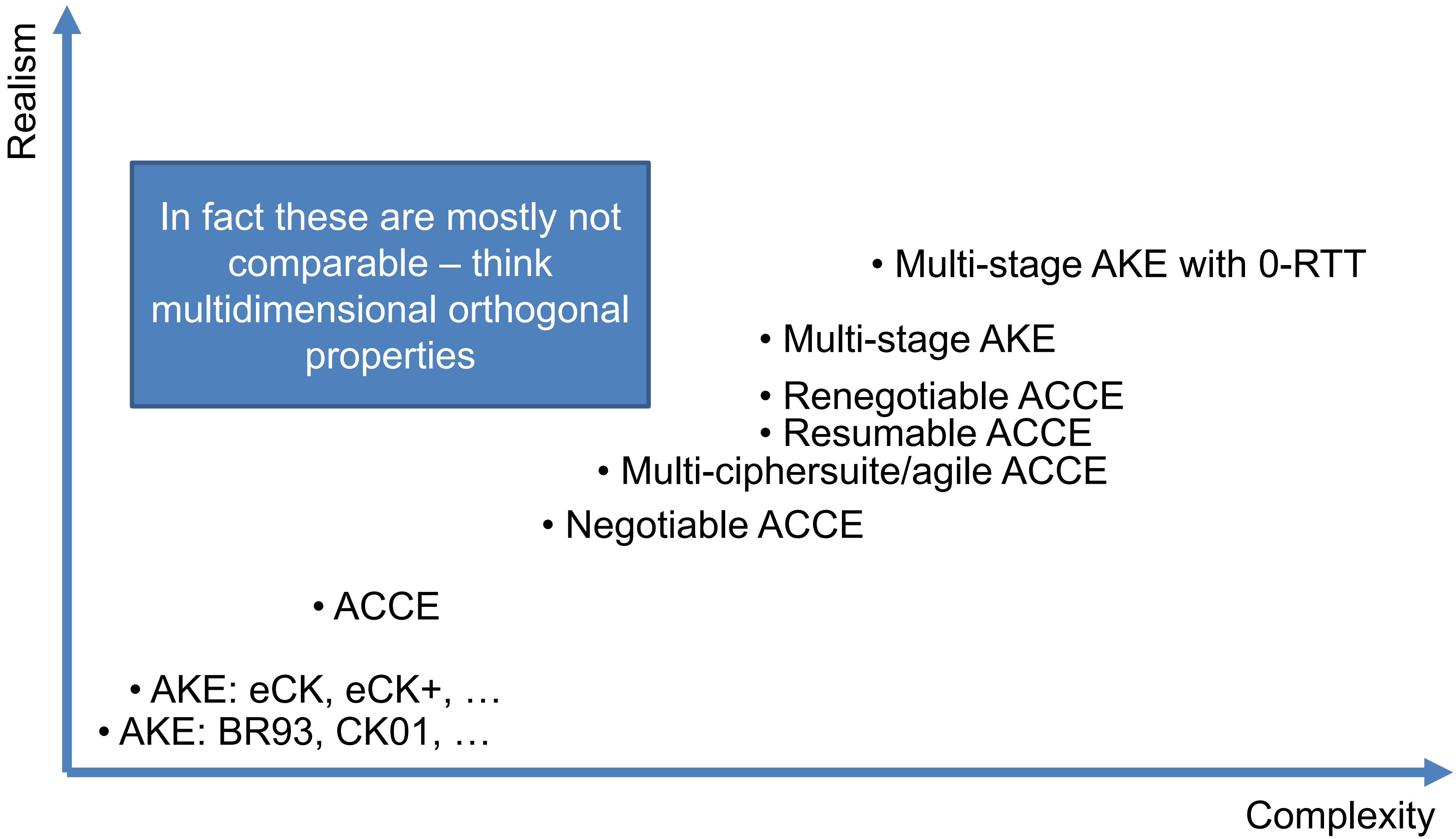
Realism

- Partially-specified authenticated encryption for streams
- Authenticated encryption for streams
- Buffered stateful authenticated encryption
- Stateful length-hiding authenticated encryption
- Stateful authenticated encryption
- Authenticated encryption
- Confidentiality
- Integrity

Complexity

Inspired by Tom Shrimpton's talk yesterday

AKE & ACCE-like models



Directions for models

- Multi-ciphersuite negotiable renegotiable multi-stage 0-RTT-capable stateful length-hiding streaming partially-specified authenticated and confidential channel establishment

We're done!

- There won't be "one true model"
  - Develop models to assess particular characteristics of a protocol
  - Often the model will be tailored specifically to the protocol in question
  - Try to supply composability and modularity where possible to tame complexity
  - Doesn't provide a satisfying "full" treatment of a protocol

Other analysis approaches

Abstract / constructive cryptography

- Focuses on constructing protocols by composing building blocks

- [Tackmann PhD thesis 2014]
- [Badertscher et al. ProvSec 2015]
- [Kohlweiss et al. INDOCRYPT 2015]

Automated model checking

- Uses verification tools to check that protocols cannot enter prohibited states
- Useful for complex protocol interactions

- [Cremers et al. S&P 2016]
  - Checked interaction between PSK and full TLS 1.3 handshakes

Other analysis approaches

Formal methods

- Additional work on model checkers
  - [Mitchell et al. Usenix 98] to SSLv2/v3
- Theorem provers
  - [Paulson ACM TISSEC Aug 99]
  - [Ogato and Futatsugi ICDCS 2005]
- Logic-based proofs
  - [He et al. CCS 2005]
  - [Kamil & Lowe JCS Sep 2011]

Formal methods with verified implementations

- Formally specified model along with implementation that is verified to meet the specification

- [Jürjens ASE 2006] – SSL in Java
- [Chaki and Data CSF 2009] - OpenSSL

- miTLS and Everest projects
  - https://mitls.org/
  - https://project-everest.github.io/

Other protocols

Internet protocols

- DNSSEC
- Internet Key Exchange (IKE)
- IPsec
- Noise framework
- NTP
- Pond
- Signal
- Tor
- Wireguard

Broader protocols

- DOCSIS (cable boxes)
- EMV (Chip & Pin)
- ePassports

- Industrial control systems
- Internet of Things
- Mobile phones
- Vehicle networks

Provable security of Internet cryptography protocols

Douglas Stebila

McMaster University

UNIVERSITY OF WATERLOO

Based on joint works with Florian Bergsma, Katriel Cohn-Gordon, Cas Cremers, Ben Dowling, Marc Fischlin, Felix Günther, Luke Garratt, Florian Kohlar, Jörg Schwenk

Funding acknowledgements:
ATN-DAAD, ARC

NSERC
CRSNG

Summer School on real-world crypto and privacy • Šibenik, Croatia • June 15, 2018
https://www.douglas.stebila.ca/research/presentations