Provable security of advanced properties of TLS and SSH

Douglas Stebila

joint work with Ben Dowling (QUT), Florian Giesen, Florian Kohlar, Jörg Schwenk (Bochum)

eprint 2012/630 (CCS'13), eprint 2013/813

2014/01/15
Real World Cryptography

Queensland University of Technology

Supported by:

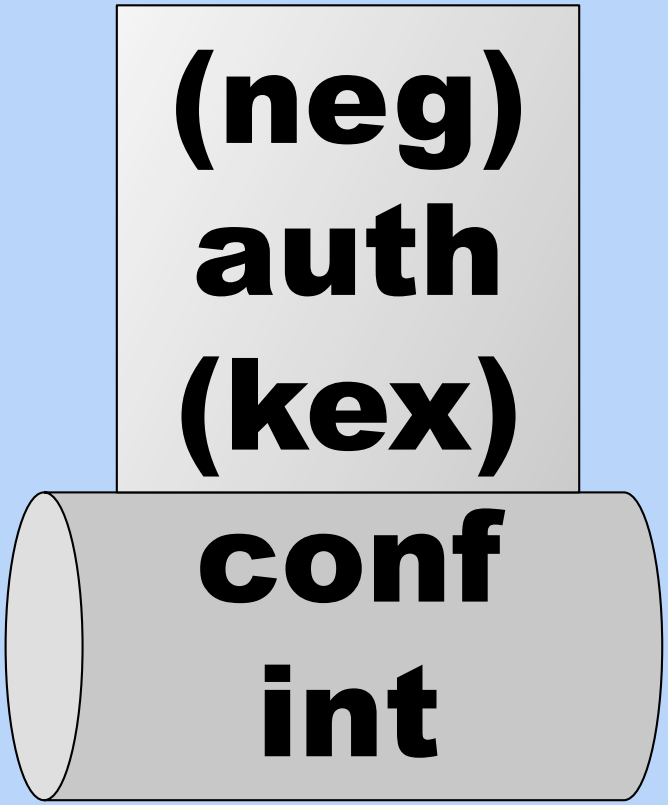
Australian Technology Network–German Academic Exchange Service (ATN-DAAD) Joint Research Cooperation Scheme

Australian Research Council Discovery Project

TLS vs. SSH

TLS	SSH
provides secure transport for many applications	provides secure transport primarily for remote shell logins
entity authentication	entity authentication
confidentiality & integrity of transmissions	confidentiality & integrity of transmissions
handshake establishes secure channel	handshake establishes secure channel

Security goals of TLS and SSH

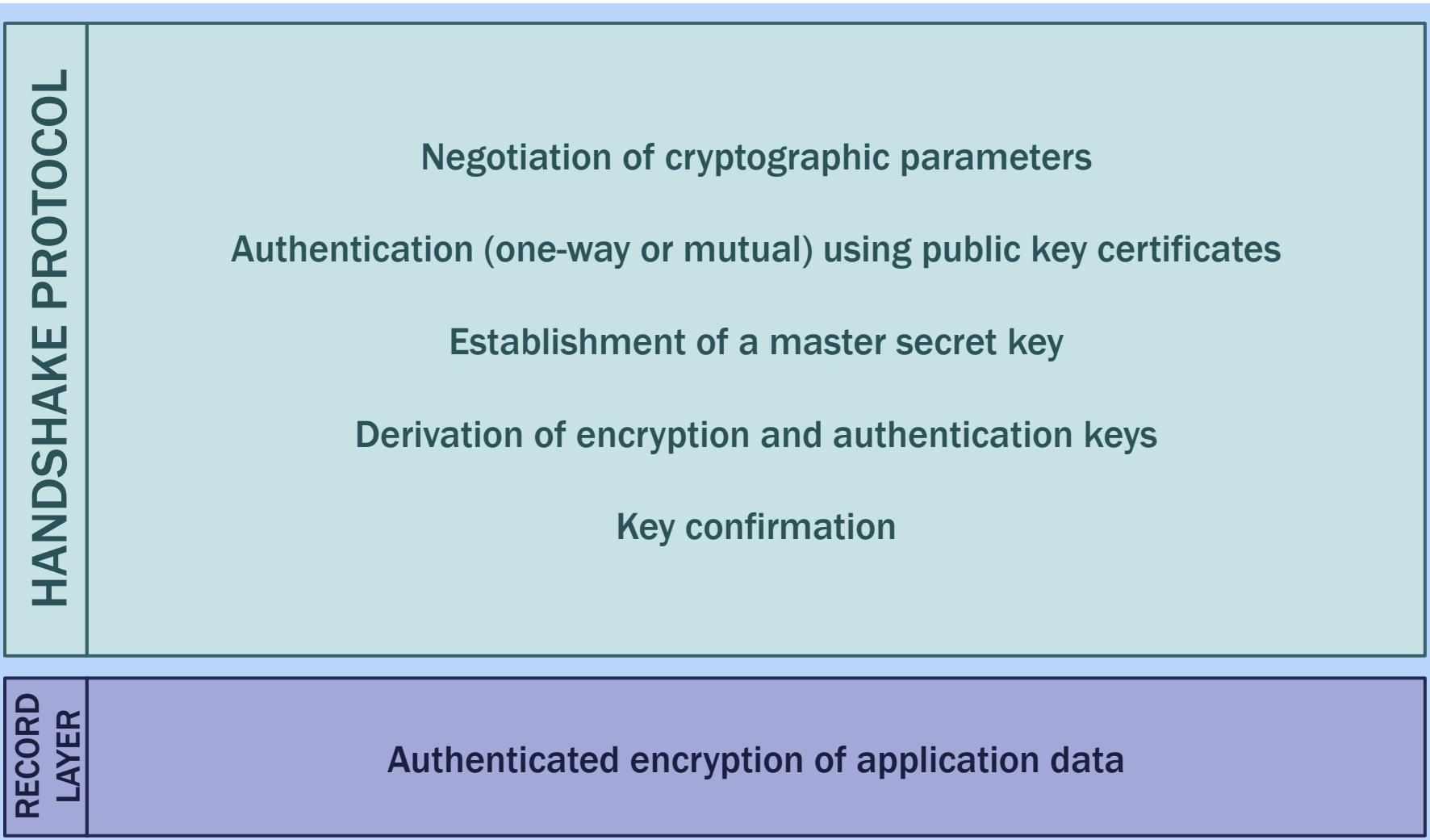


From an application perspective, TLS and SSH provide:

- (negotiation of parameters)
- entity authentication
- (key exchange)
- confidentiality and integrity of messages

Security of TLS

Structure of TLS



Components of TLS

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

- Data structures
- Key derivation
- Encryption modes, IVs
- Padding
- Compression

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

- OpenSSL
- GnuTLS
- SChannel
- Java JSSE

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

Is 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

Is TLS secure?

1996 SSL v3.0 standardized

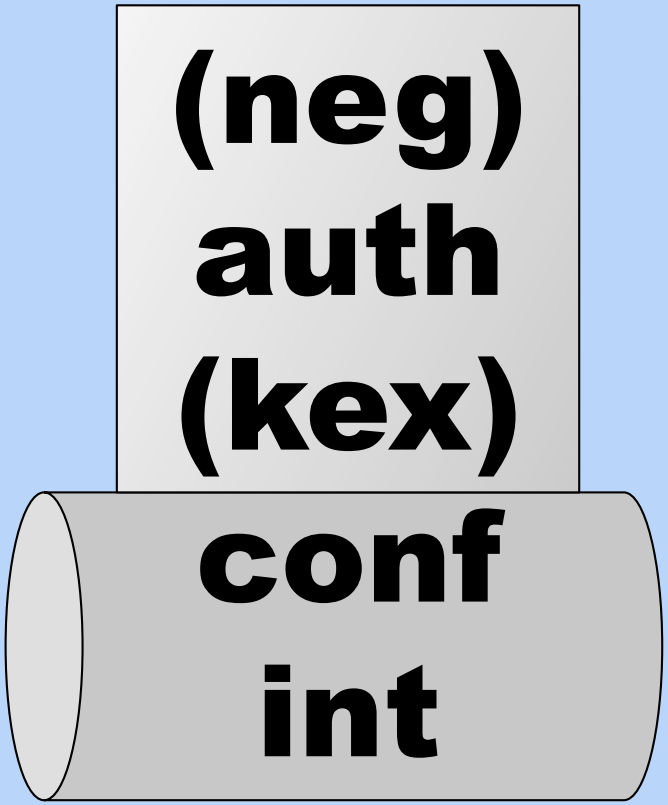
2011 Some modes of TLS record layer are secure authenticated encryption schemes [PRS11]

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"

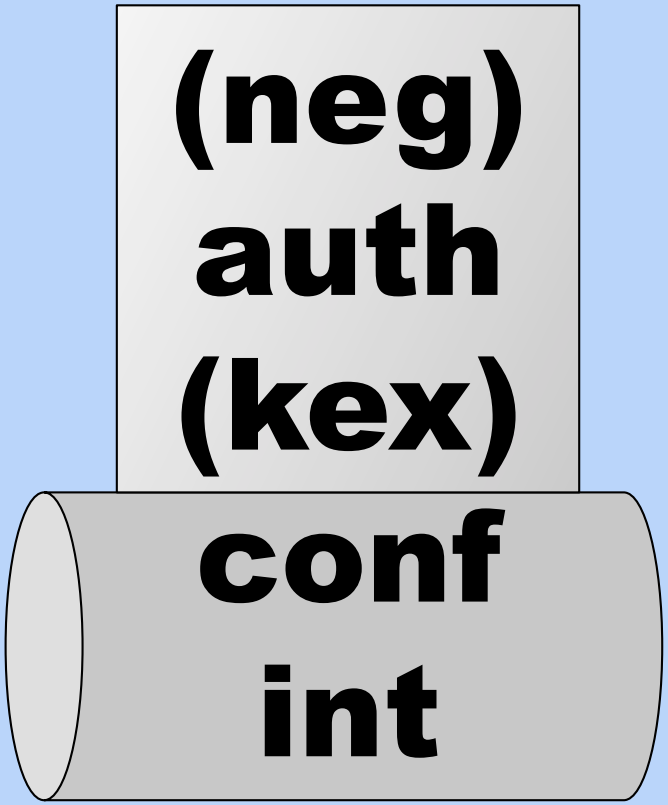
Security goals of TLS and SSH



From an application perspective, TLS and SSH provide:

- (negotiation of parameters)
- entity authentication
- (key exchange)
- confidentiality and integrity of messages

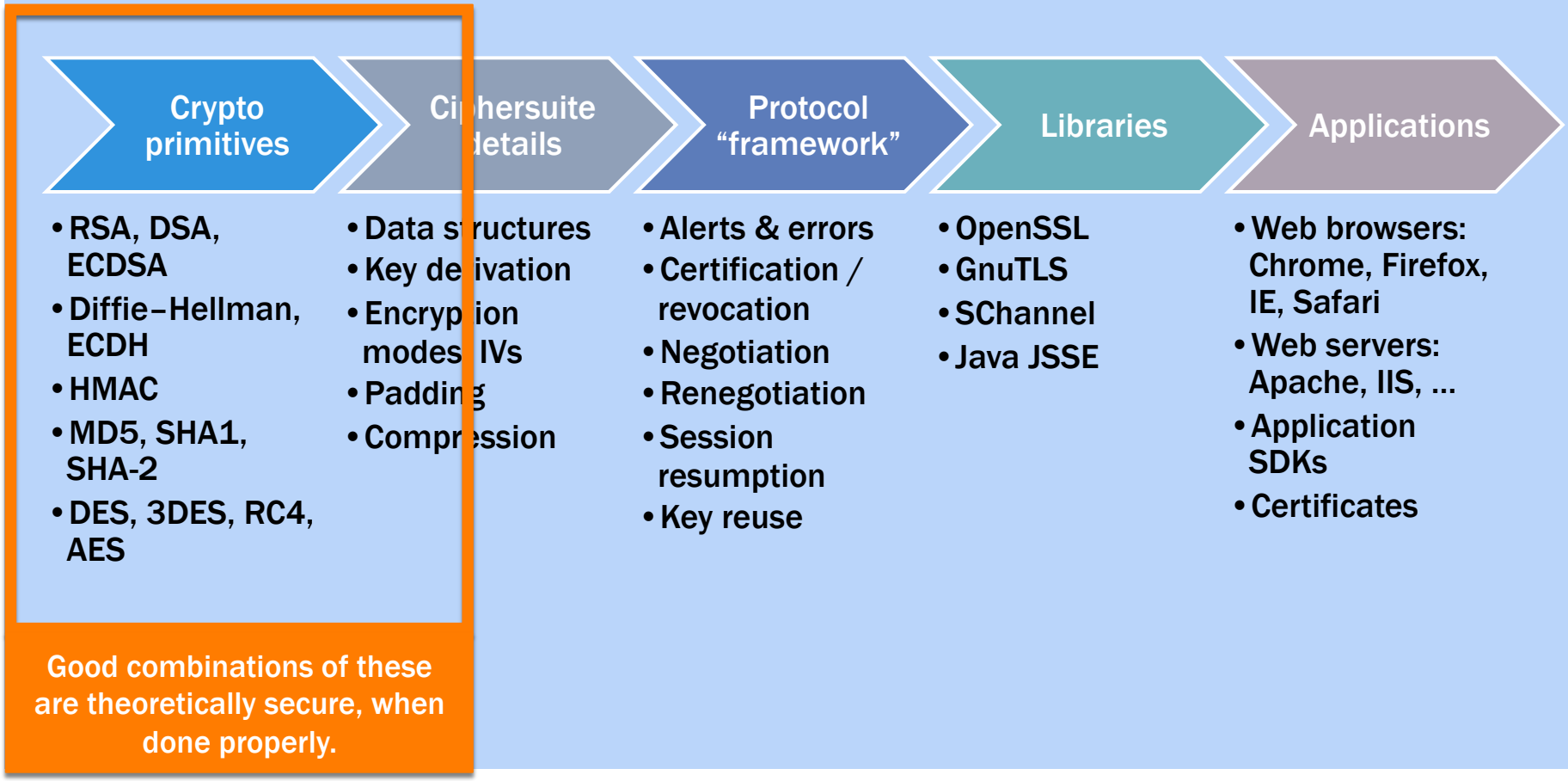
Security goals of TLS and SSH



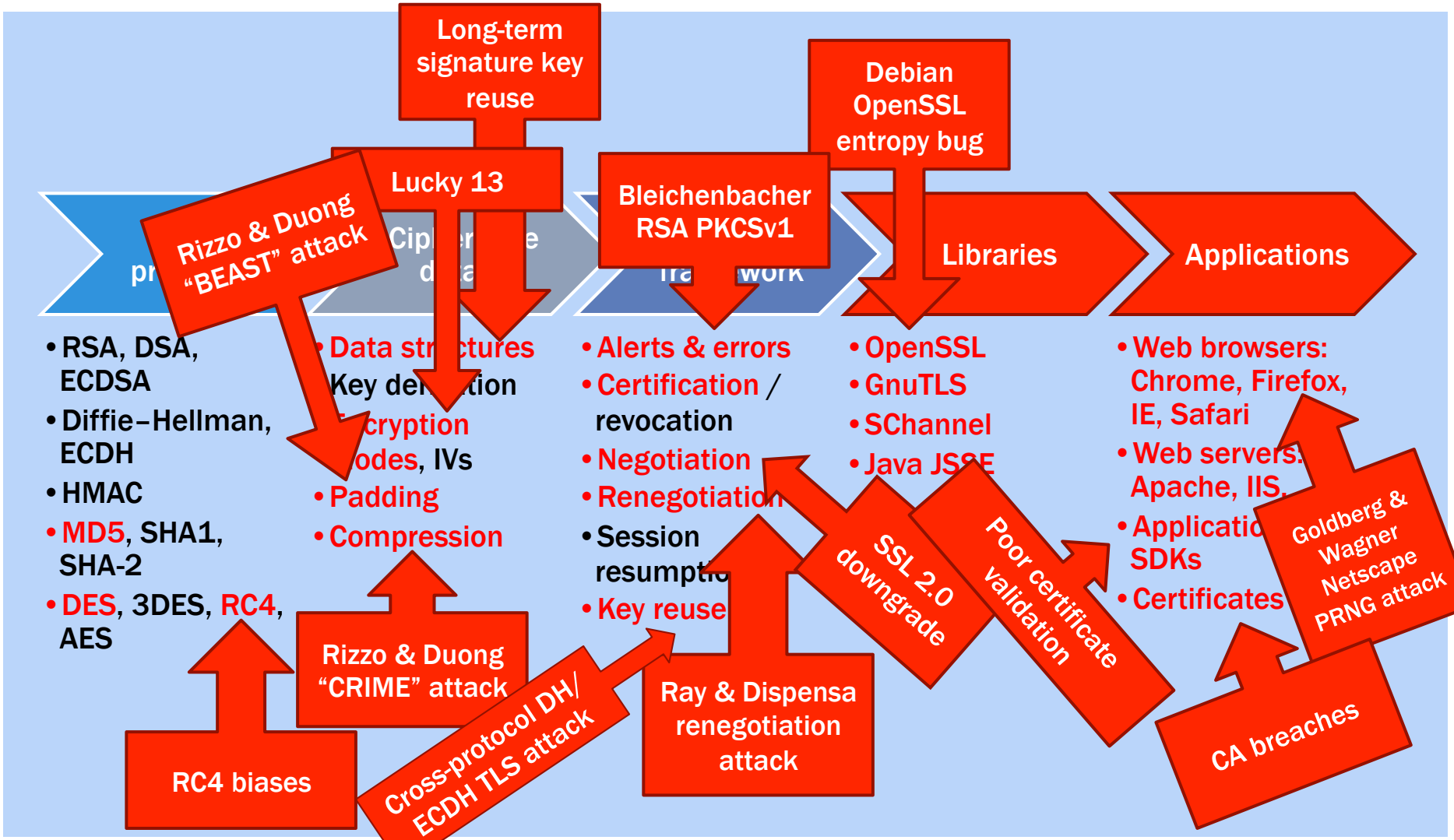
Authenticated and Confidential Channel Establishment (ACCE) security definition [JKSS12] captures:

- entity authentication
- (key exchange)
- confidentiality and integrity of messages

Results on the provable security of TLS

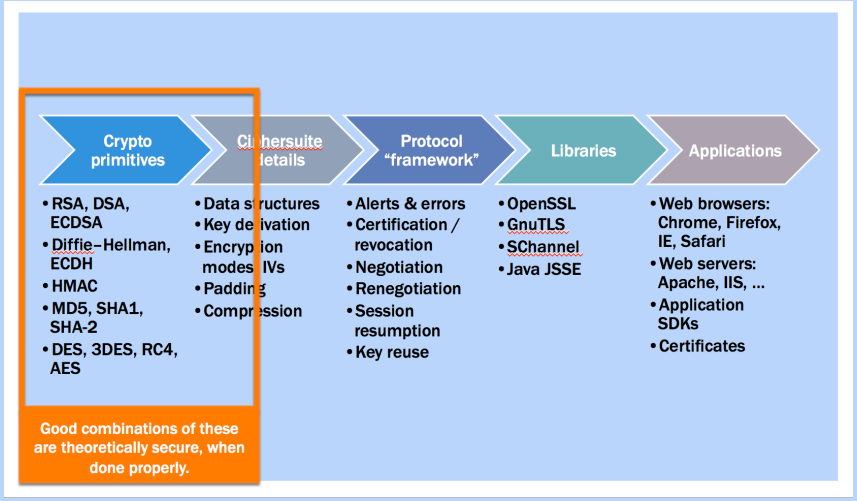


Real-world attacks on TLS

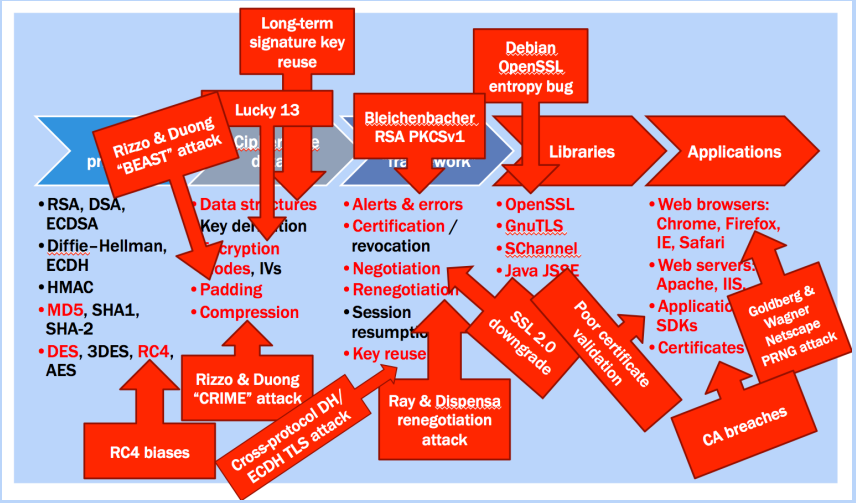


The gap between theory and practice

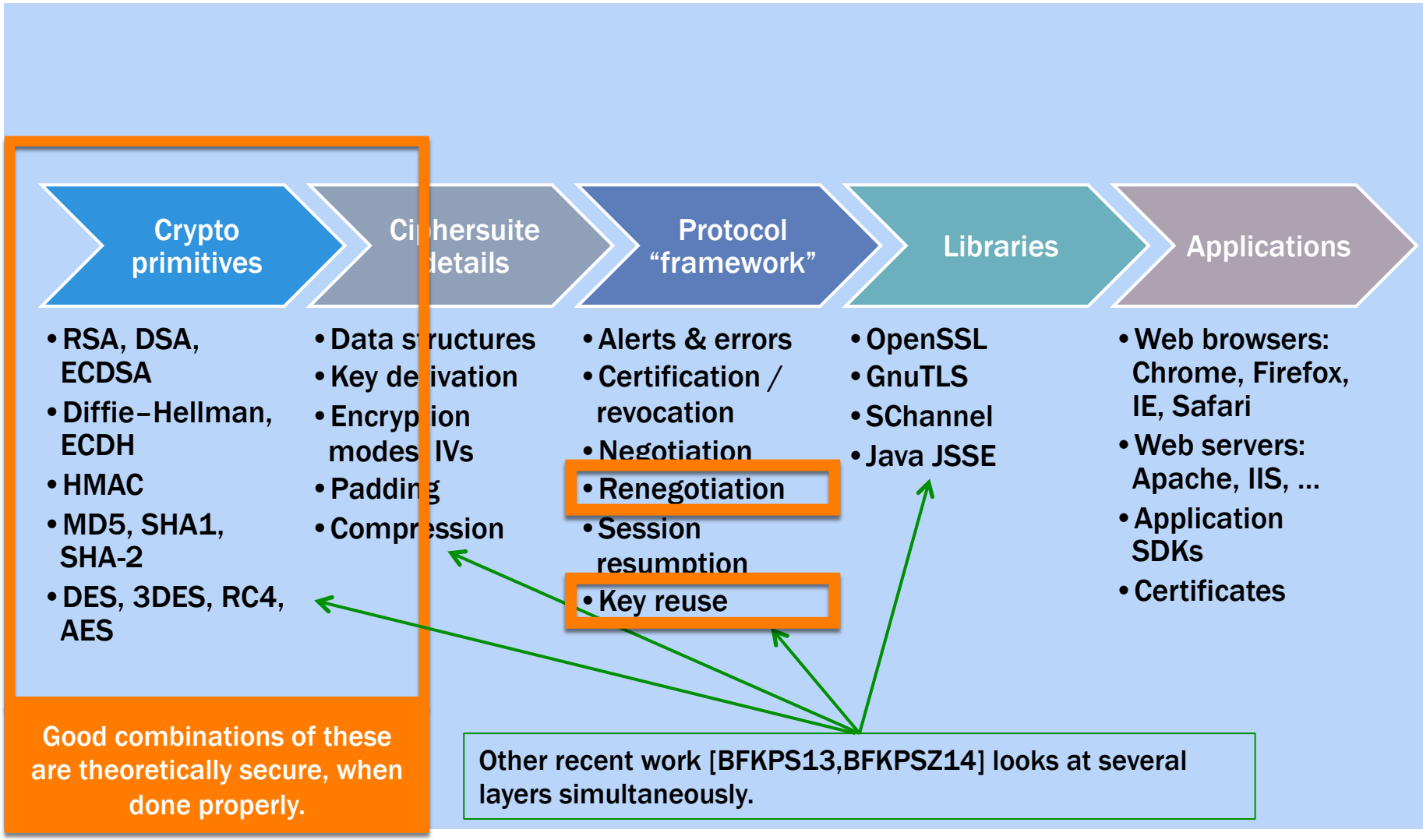
Provable security results



Real-world attacks



Components of TLS



TLS and renegotiation

ACM CCS 2013

eprint 2012/630

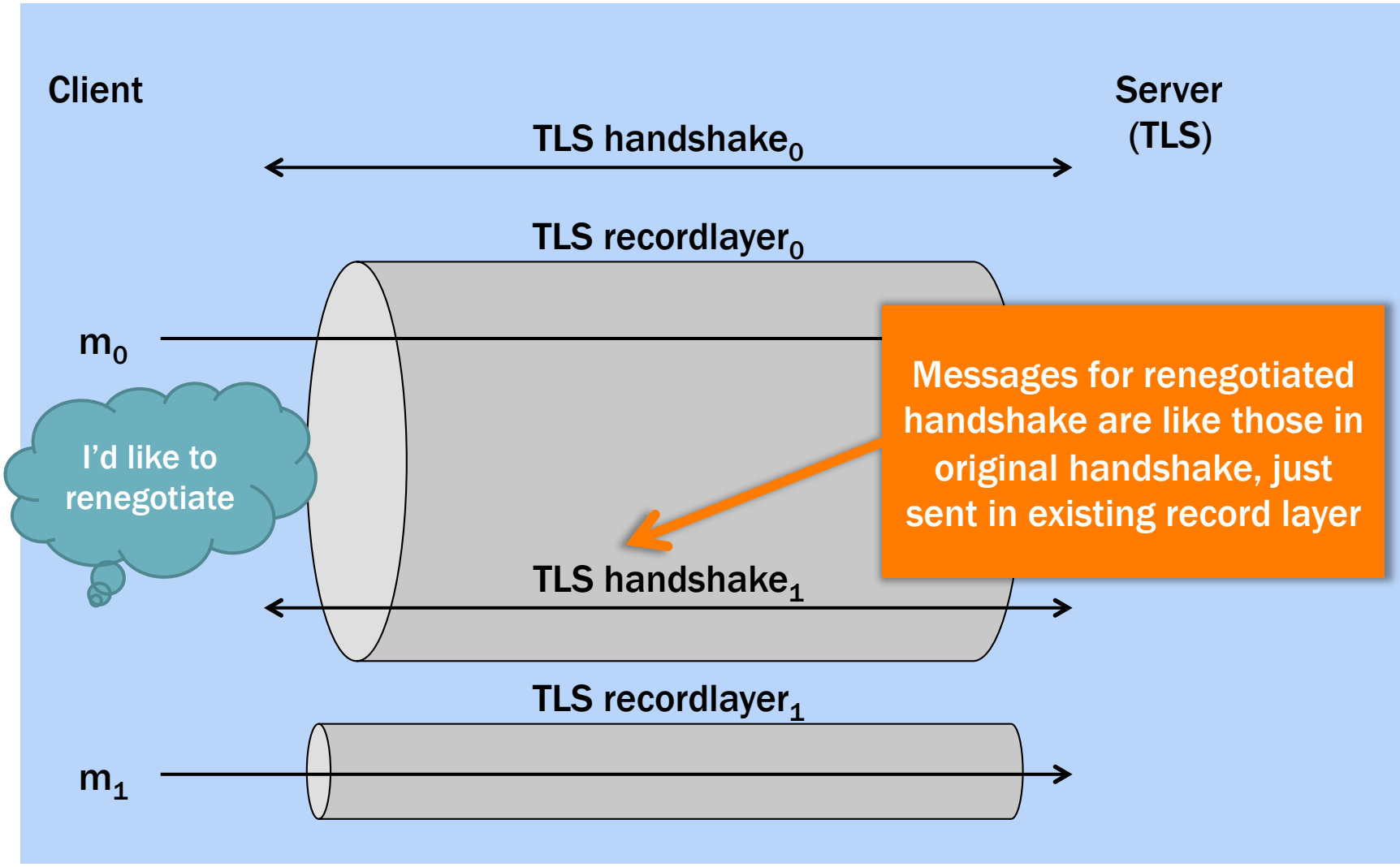
Why renegotiate?

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?

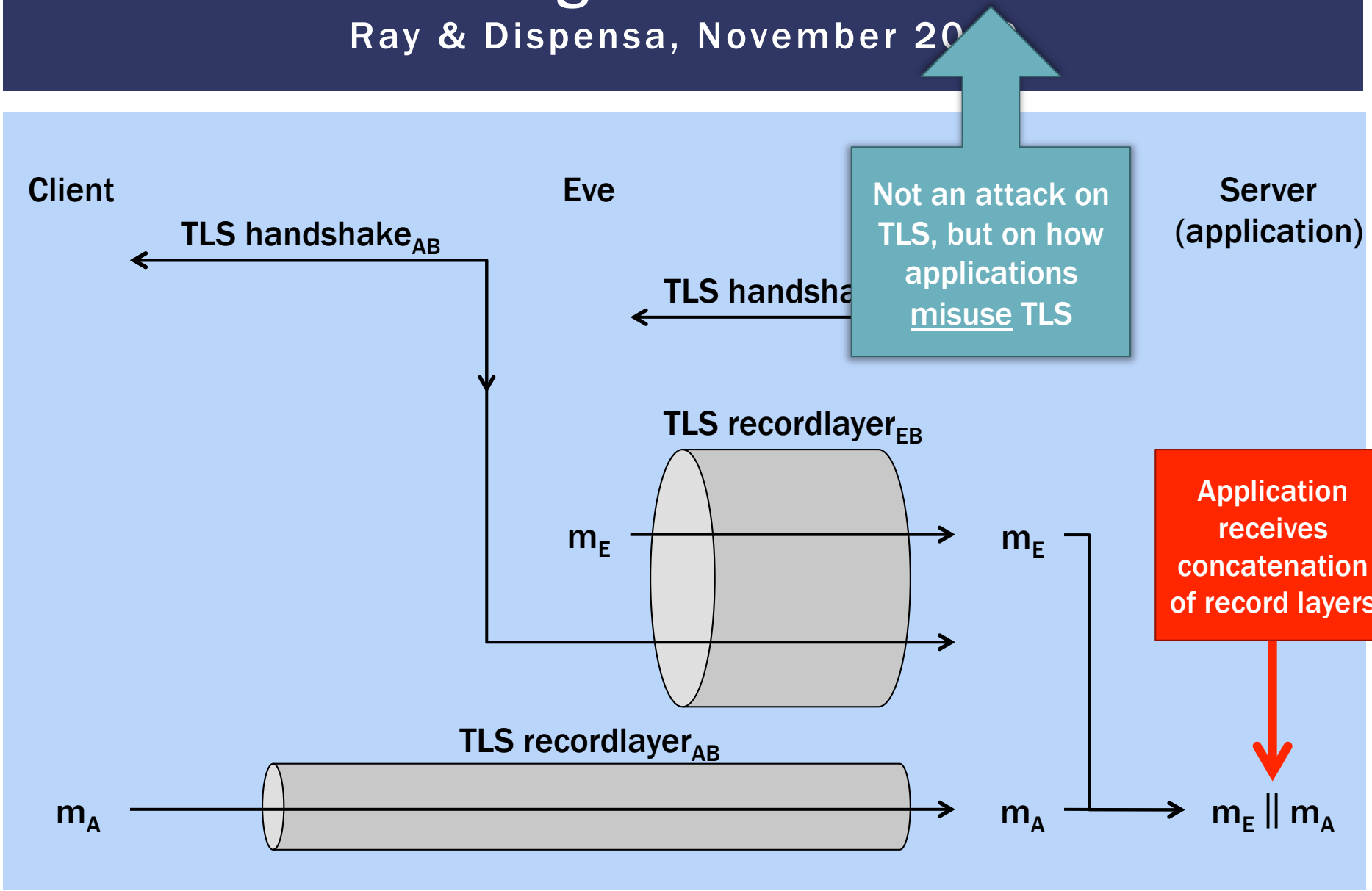
- Change cryptographic parameters
- Refresh encryption keys ("more forward secrecy")
- Change authentication credentials
- Identity hiding for client
  1. Establish a one-way authenticated TLS session
  2. Renegotiate using mutual authentication.
     Since handshake messages are sent in the encrypted TLS channel, client's identity is kept private.

Renegotiation in TLS
(pre-November 2009)



TLS Renegotiation "Attack"

Ray & Dispensa, November 20



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.

TLS Renegotiation Countermeasures

Two related countermeasures standardized by IETF in RFC 5746:

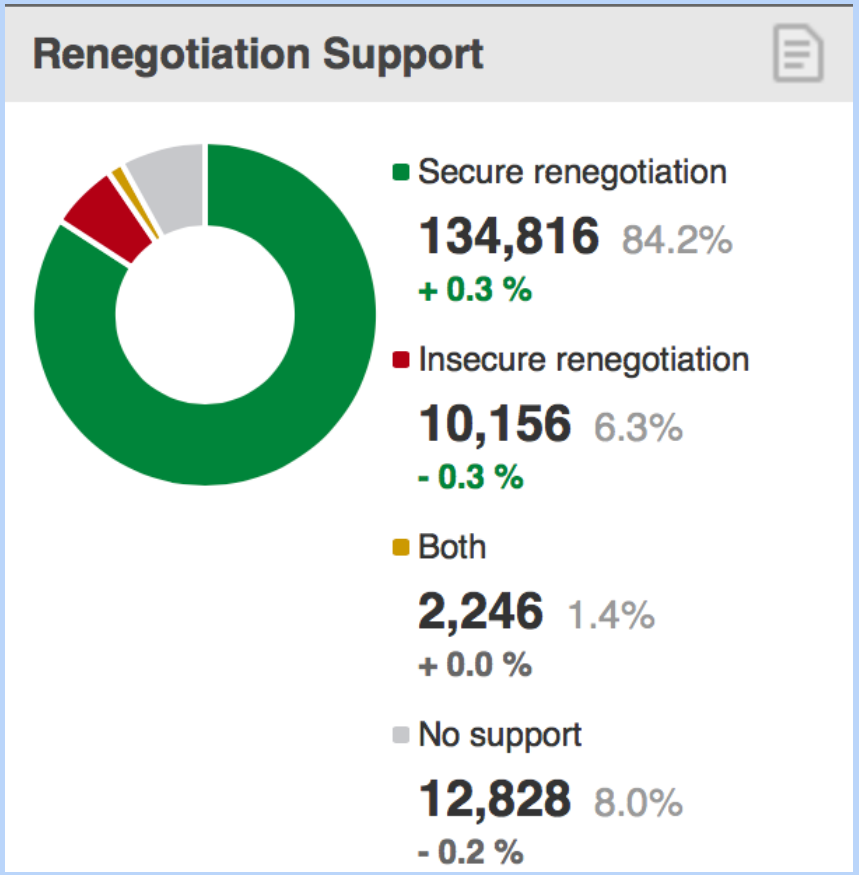
1. Signalling Ciphersuite Value
2. Renegotiation Indication Extension

Basic idea: include fingerprint of previous handshake when renegotiating.

TLS Renegotiation Countermeasures

SCSV/RIE fairly quickly and widely adopted.

Currently 84% deployment
(SSL Pulse, Jan. 3, 2014)



Does this really fix the problem?

Does this really fix the problem?

ACCE security isn't enough: these ciphersuites have been proven ACCE security yet are vulnerable to renegotiation attack.

Need a security definition that includes renegotiation.

Technical approach

1. Extend authenticated and confidential channel establishment (ACCE) security model to include renegotiable, multi-phase protocols.

2. Define security notion for renegotiable protocols.
   - weakly secure renegotiable ACCE
   - secure renegotiable ACCE

3. Show that TLS without fixes does not satisfy security definition.

4. Show that TLS with fixes does satisfy security definition.
   - Generic reduction: If a TLS ciphersuite satisfies a certain property, then, when combined with fixes, it is a weakly secure renegotiable ACCE.

5. Propose even stronger fix.

Secure renegotiable ACCE

Definition	TLS
When a party successfully renegotiates a new phase, its partner has a phase with a matching handshake and record layer transcript	▪ TLS with or without RFC 5746 fixes is not a secure renegotiable ACCE. ▪ Can be fixed by including hash of previous record layer messages in RIE.

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 good enough.)

Lessons learned: TLS renegotiation

Theory

- Renegotiation not previously included in AKE security definitions.
  - Different levels of renegotiation security
- Security of a protocol in isolation doesn't imply security with renegotiation.
- Need to "open up" security definitions in order to generically compose protocols.

Practice

- Confidence in standardized TLS renegotiation fixes.

Multi-ciphersuite security, TLS and SSH

eprint 2013/813

TLS_NULL_WITH_NULL_NULL TLS_RSA_WITH_NULL_MD5 TLS_RSA_WITH_NULL_SHA TLS_RSA_EXPORT_WITH_RC4_40_MD5 TLS_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 TLS_RSA_WITH_IDEA_CBC_SHA TLS_RSA_EXPORT_WITH_DES40_CBC_SHA TLS_RSA_WITH_DES_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA TLS_DH_DSS_WITH_DES_CBC_SHA TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA TLS_DH_RSA_WITH_DES_CBC_SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA TLS_DHE_DSS_WITH_DES_CBC_SHA TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA TLS_DH_anon_EXPORT_WITH_RC4_40_MD5 TLS_DH_anon_WITH_RC4_128_MD5 TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_WITH_DES_CBC_SHA TLS_DH_anon_WITH_3DES_EDE_CBC_SHA TLS_KRB5_WITH_DES_CBC_SHA TLS_KRB5_WITH_3DES_EDE_CBC_SHA TLS_KRB5_WITH_RC4_128_SHA TLS_KRB5_WITH_IDEA_CBC_SHA
TLS_KRB5_WITH_DES_CBC_MD5 TLS_KRB5_WITH_3DES_EDE_CBC_MD5 TLS_KRB5_WITH_RC4_128_MD5 TLS_KRB5_WITH_IDEA_CBC_MD5 TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA
TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA TLS_KRB5_EXPORT_WITH_RC4_40_SHA TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5 TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5 TLS_KRB5_EXPORT_WITH_RC4_40_MD5
TLS_PSK_WITH_NULL_SHA TLS_DHE_PSK_WITH_NULL_SHA TLS_RSA_PSK_WITH_NULL_SHA TLS_RSA_WITH_AES_128_CBC_SHA TLS_DH_DSS_WITH_AES_128_CBC_SHA TLS_DH_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_DH_anon_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA TLS_DH_DSS_WITH_AES_256_CBC_SHA
TLS_DH_RSA_WITH_AES_256_CBC_SHA TLS_DHE_DSS_WITH_AES_256_CBC_SHA TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLS_DH_anon_WITH_AES_256_CBC_SHA TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_DH_DSS_WITH_AES_128_CBC_SHA256 TLS_DH_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DH_DSS_WITH_AES_256_CBC_SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA256 TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 TLS_DH_anon_WITH_AES_128_CBC_SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA TLS_PSK_WITH_RC4_128_SHA TLS_PSK_WITH_3DES_EDE_CBC_SHA
TLS_PSK_WITH_AES_128_CBC_SHA TLS_PSK_WITH_AES_256_CBC_SHA TLS_DHE_PSK_WITH_RC4_128_SHA TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA TLS_DHE_PSK_WITH_AES_128_CBC_SHA
TLS_DHE_PSK_WITH_AES_256_CBC_SHA TLS_RSA_PSK_WITH_RC4_128_SHA TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA TLS_RSA_PSK_WITH_AES_128_CBC_SHA TLS_RSA_PSK_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_SEED_CBC_SHA TLS_DH_DSS_WITH_SEED_CBC_SHA TLS_DH_RSA_WITH_SEED_CBC_SHA TLS_DHE_DSS_WITH_SEED_CBC_SHA TLS_DHE_RSA_WITH_SEED_CBC_SHA TLS_DH_anon_WITH_SEED_CBC_SHA
TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 TLS_DH_RSA_WITH_AES_128_GCM_SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 TLS_DH_DSS_WITH_AES_128_GCM_SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384 TLS_DH_anon_WITH_AES_128_GCM_SHA256 TLS_DH_anon_WITH_AES_256_GCM_SHA384 TLS_PSK_WITH_AES_128_GCM_SHA256 TLS_PSK_WITH_AES_256_GCM_SHA384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 TLS_RSA_PSK_WITH_AES_256_GCM_SHA384
TLS_PSK_WITH_AES_128_CBC_SHA256 TLS_PSK_WITH_AES_256_CBC_SHA384 TLS_PSK_WITH_NULL_SHA256 TLS_PSK_WITH_NULL_SHA384 TLS_DHE_PSK_WITH_AES_128_CBC_SHA256
TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 TLS_DHE_PSK_WITH_NULL_SHA256 TLS_DHE_PSK_WITH_NULL_SHA384 TLS_RSA_PSK_WITH_AES_128_CBC_SHA256 TLS_RSA_PSK_WITH_AES_256_CBC_SHA384
TLS_RSA_PSK_WITH_NULL_SHA256 TLS_RSA_PSK_WITH_NULL_SHA384 TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256 TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256 TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256 TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256 TLS_EMPTY_RENEGOTIATION_INFO_SCSV TLS_ECDH_ECDSA_WITH_NULL_SHA TLS_ECDH_ECDSA_WITH_RC4_128_SHA TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_NULL_SHA TLS_ECDHE_ECDSA_WITH_RC4_128_SHA TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDH_RSA_WITH_NULL_SHA TLS_ECDH_RSA_WITH_RC4_128_SHA TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_NULL_SHA TLS_ECDHE_RSA_WITH_RC4_128_SHA TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDH_anon_WITH_NULL_SHA TLS_ECDH_anon_WITH_RC4_128_SHA TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_anon_WITH_AES_128_CBC_SHA TLS_ECDH_anon_WITH_AES_256_CBC_SHA TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA
TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA TLS_SRP_SHA_WITH_AES_128_CBC_SHA TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA
TLS_SRP_SHA_WITH_AES_256_CBC_SHA TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_PSK_WITH_RC4_128_SHA
TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 TLS_ECDHE_PSK_WITH_NULL_SHA TLS_ECDHE_PSK_WITH_NULL_SHA256 TLS_ECDHE_PSK_WITH_NULL_SHA384 TLS_RSA_WITH_ARIA_128_CBC_SHA256
TLS_RSA_WITH_ARIA_256_CBC_SHA384 TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256 TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384 TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256 TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384
TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256 TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384 TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384
TLS_DH_anon_WITH_ARIA_128_CBC_SHA256 TLS_DH_anon_WITH_ARIA_256_CBC_SHA384 TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384
TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384 TLS_RSA_WITH_ARIA_128_GCM_SHA256 TLS_RSA_WITH_ARIA_256_GCM_SHA384
TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384 TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256 TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384
TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256 TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384 TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256 TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384
TLS_DH_anon_WITH_ARIA_128_GCM_SHA256 TLS_DH_anon_WITH_ARIA_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384
TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384 TLS_PSK_WITH_ARIA_128_CBC_SHA256 TLS_PSK_WITH_ARIA_256_CBC_SHA384
TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256 TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384 TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256 TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384
TLS_PSK_WITH_ARIA_128_GCM_SHA256 TLS_PSK_WITH_ARIA_256_GCM_SHA384 TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256 TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384
TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256 TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384 TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256 TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384 TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384 TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384 TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384 TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256
TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384 TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256 TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384 TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256
TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384 TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256 TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384
TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256 TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384 TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256 TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384
TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256 TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384 TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384
TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256 TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384 TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384
TLS_RSA_WITH_AES_128_CCM TLS_RSA_WITH_AES_256_CCM TLS_DHE_RSA_WITH_AES_128_CCM TLS_DHE_RSA_WITH_AES_256_CCM TLS_RSA_WITH_AES_128_CCM_8 TLS_RSA_WITH_AES_256_CCM_8
TLS_DHE_RSA_WITH_AES_128_CCM_8 TLS_DHE_RSA_WITH_AES_256_CCM_8 TLS_PSK_WITH_AES_128_CCM TLS_PSK_WITH_AES_256_CCM TLS_DHE_PSK_WITH_AES_128_CCM TLS_DHE_PSK_WITH_AES_256_CCM
TLS_PSK_WITH_AES_128_CCM_8 TLS_PSK_WITH_AES_256_CCM_8 TLS_PSK_DHE_WITH_AES_128_CCM_8 TLS_PSK_DHE_WITH_AES_256_CCM_8

List of all 314 TLS ciphersuites

List of SSH ciphersuites

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

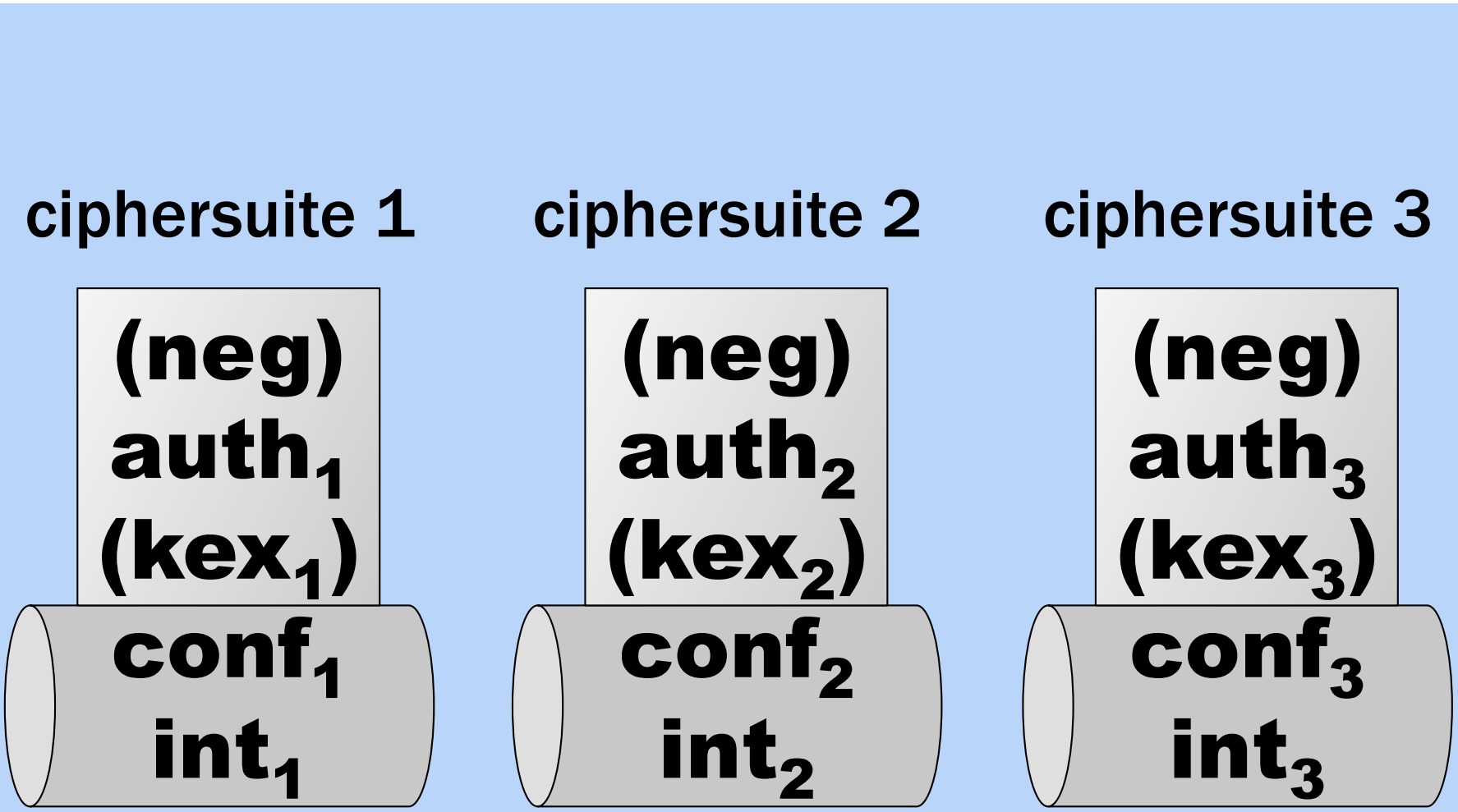
Encryption:

- 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

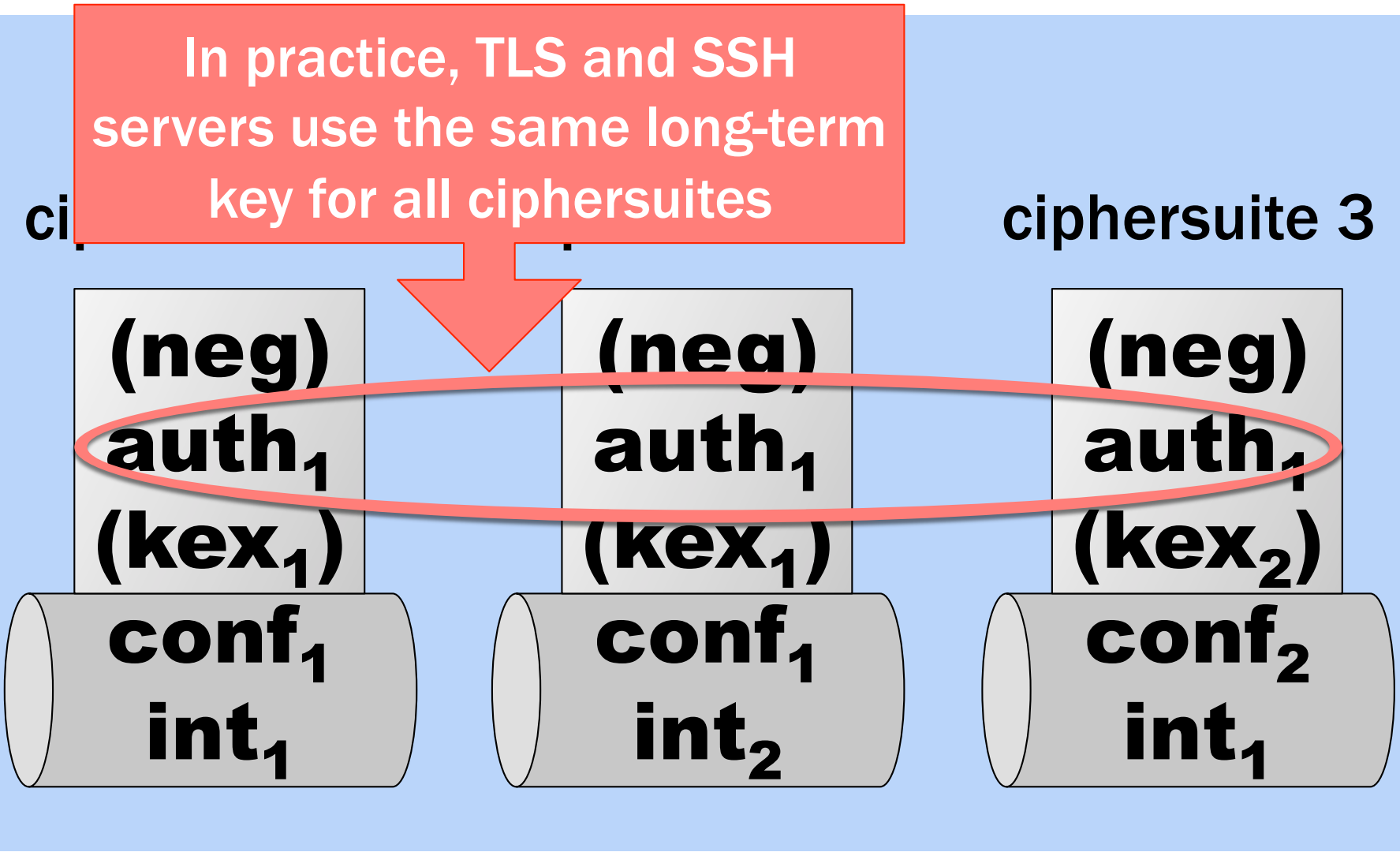
MACs:

- hmac-sha1
- hmac-sha1-96
- hmac-md5
- hmac-md5-96
- AEAD_AES_128_GCM
- AEAD_AES_256_GCM
- hmac-sha2-256
- hmac-sha2-512

How we'd like to analyze ciphersuites



The reality of multi-ciphersuite usage



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.

Long-term keys in TLS

Most TLS ciphersuites are provably secure channels (ACCE).

But this assumes that each ciphersuite uses its own distinct long-term key.

Long-term keys in TLS

In signed Diffie–Hellman ciphersuites, the thing that is signed is the raw DH parameters.

- i.e., without type information

Possible to interpret a set of ECDH params as valid finite field DH params.

- [MVVP12]

=> TLS not secure with long-term key reuse.

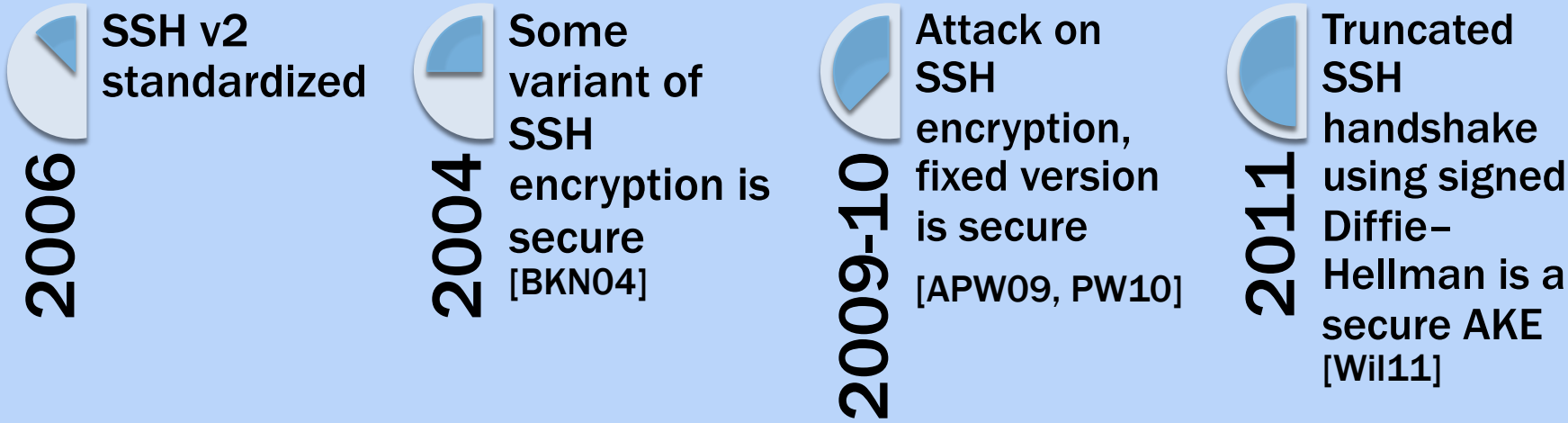
ACCE security of a ciphersuite in isolation does not imply security with long-term key reuse.

Long-term keys in SSH

In SSH, the thing that is signed contains an unambiguous identification of the intended ciphersuite.

We might hope to be able to prove SSH secure even with key reuse across ciphersuites.

Is SSH secure?



"some variant"… "truncated SSH"

Signed DH SSH is a secure ACCE

Theorem: Assuming the signature scheme is secure, the computational Diffie–Hellman problem is hard, and the encryption scheme is a secure buffered stateful authenticated encryption scheme, then signed-DH SSH is a secure ACCE protocol.

How can we prove it secure even with long-term key reuse across ciphersuites?

Provable security of long-term key reuse

Goal: Generic theorem: If an individual ciphersuite is secure, then it is secure even if long-term keys are reused across ciphersuites.

- Impossible: TLS attack from previous slide.

Revised goal: Generic theorem: If an individual ciphersuite is secure under additional conditions, then it is secure even if long-term keys are reused across ciphersuites.

Lessons learned: multi-ciphersuite

Theory

- Definition for security of multi-ciphersuite protocols.
- Generic theorem on when it is safe to reuse long-term keys across individually secure ciphersuites.
  - Main idea: adding an auxiliary “signing oracle” to individual security to enable reduction, parameterize freshness condition.
  - Lots of other applications of this main idea…

Practice

- Confidence in signed-DH SSH ciphersuites, even if the same long-term keys are reused across ciphersuites.
  - … and even when reused with unambiguously independent protocols.

Summary

Theory

- Provable security of single ciphersuites in isolation doesn't imply security in complex settings:
  - TLS renegotiation attack
  - multi-ciphersuite security
- Can augment ACCE security models for more complex functionality
- By opening up ACCE security models, can prove more generic composition theorems

Practice

- Confidence in TLS standardized renegotiation fixes.
- Confidence in SSH signed-DH ciphersuites in isolation or with long-term key reuse.

Cryptographers and standards

Tools for cryptographic modelling were not fully developed when key standards (TLS, SSH) were first proposed.

- Some flaws: padding, encryption modes, …
- Some successes: general structure adequate

Encourage cryptographers and standards community to engage to support development & adoption of protocols with formal security properties where possible.