Downgrade Resilience in Key Exchange

markulf kohlweiss

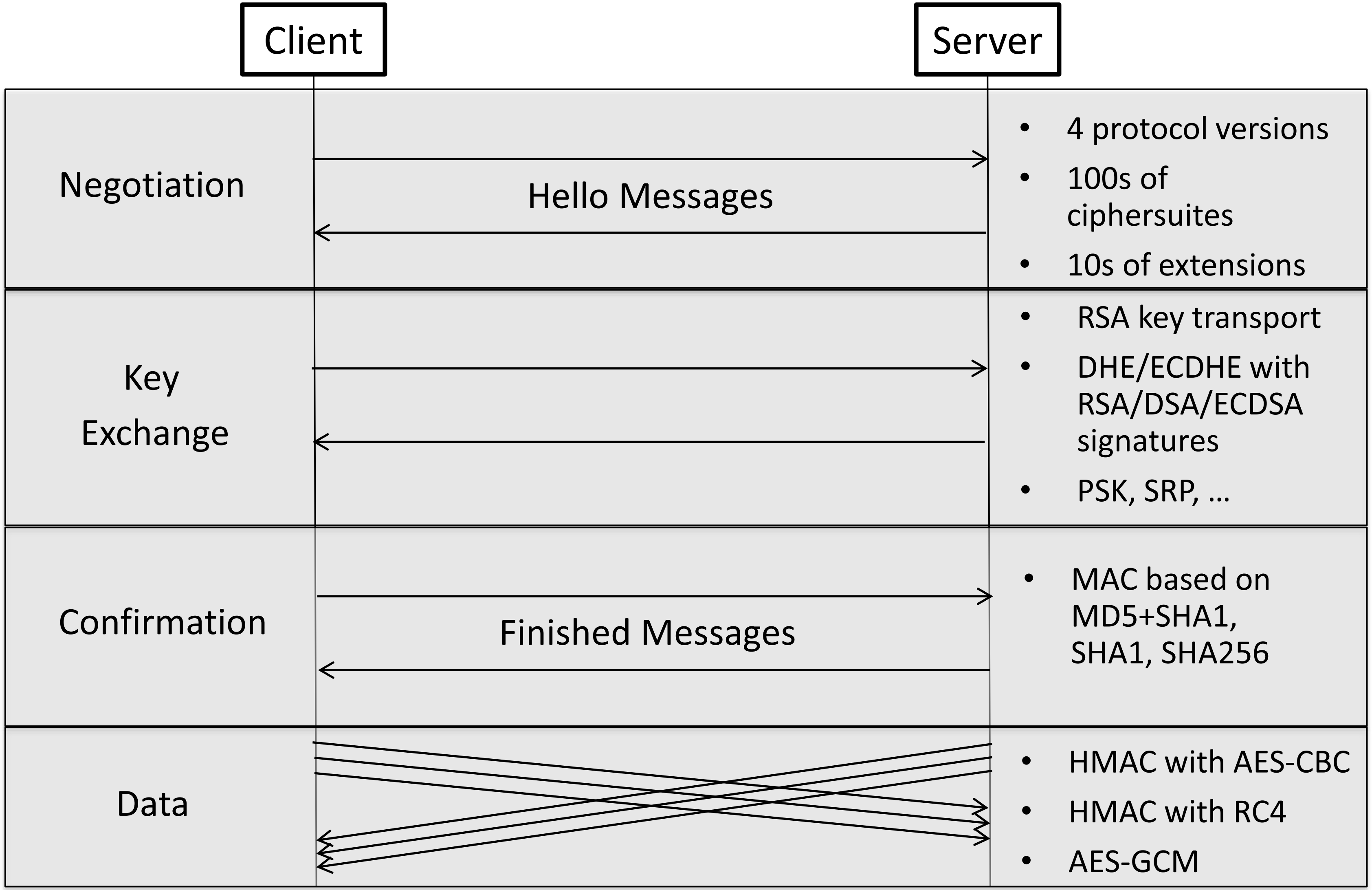
joint work with:
k. bhargavan, c. brzuska, c. fournet,
m. green, s. zanella-beguelin

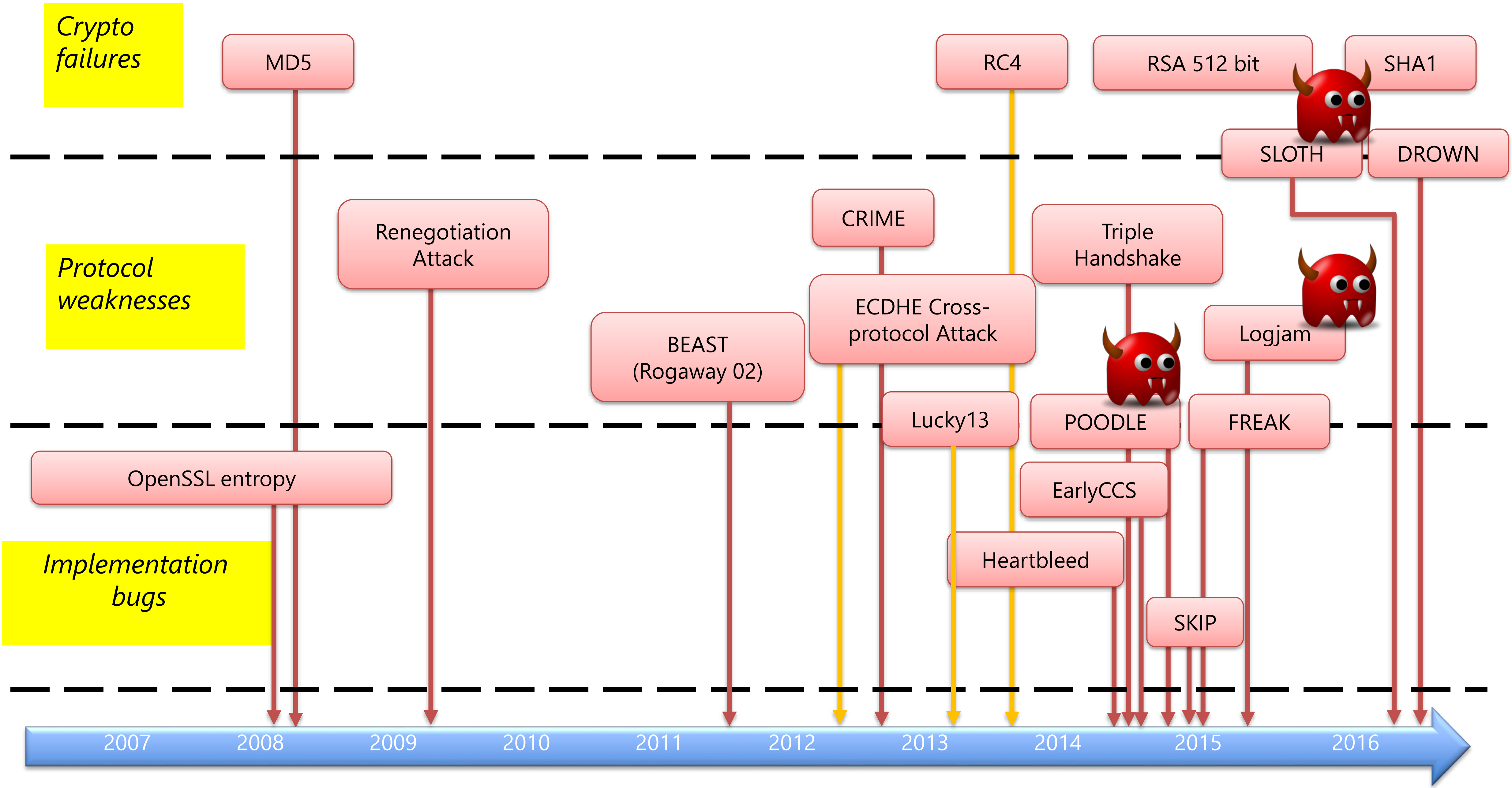
Downgrade as an everyday phenomen

https://

http://

TLS protocol suite – not a single protocol





Crypto failures
Protocol weaknesses
Implementation bugs
MD5
RC4
RSA 512 bit
SHA1
SLOTH
DROWN
Renegotiation Attack
CRIME
Triple Handshake
BEAST (Rogaway 02)
ECDHE Cross-protocol Attack
Logjam
Lucky13
POODLE
FREAK
OpenSSL entropy
EarlyCCS
Heartbleed
SKIP
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016

- POODLE
- LOGJAM
- SLOTH

Our contribution

1. Definition that tolerate weak algorithms
   – and capture downgrade attacks
2. Modular proof strategy

- Analyse downgrade security of SSH, IKE, ZRTP, **TLS**
- Prove downgrade security for SSH and **TLS 1.3**

New countermeasures designed together with core-design team of TLS 1.3

Negotiation

- Inputs:
  - $config_C$ & $config_S$: supported versions, ciphers, groups, long-term keys
- Outputs:
  - $mode$: negotiated version, cipher, group, etc.
- Ideal negotiation:
  - $mode = \mathrm{Nego}(config_C, config_S)$

Transcript authentication vs. Downgrades

- ***Authentication***
  If my negotiated *mode* uses only strong algorithms,
  then my partner and I agree on
  keys, identities and *mode.*

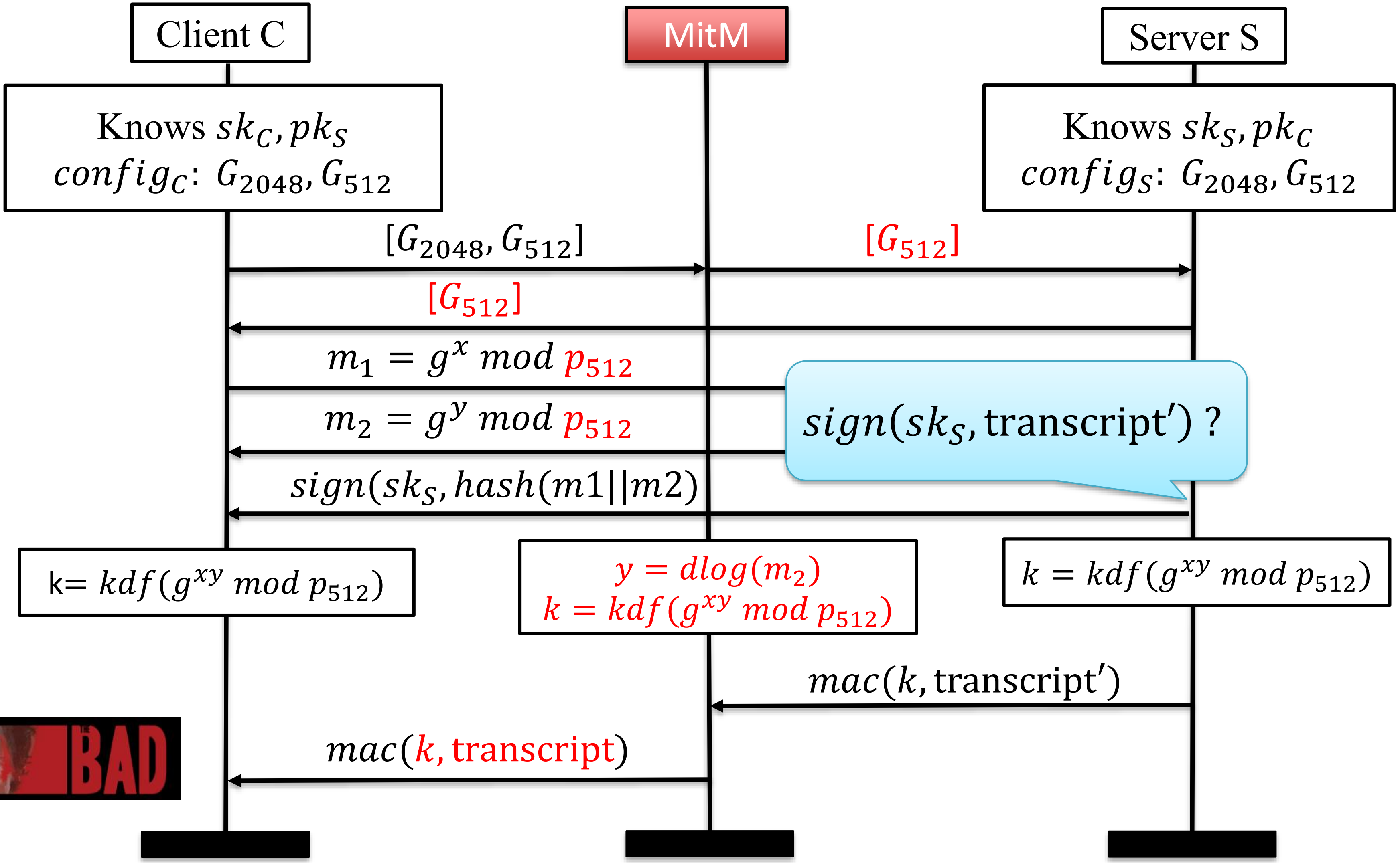
- Authentication does not guarantee
  negotiation of a strong mode.
  - **Intersection** of *config*$_C$ & *config*$_S$ must be strong!
  - What if *config*$_C$ & *config*$_S$ include a legacy algorithm?
  - What are minimal requirements on *config*$_C$ & *config*$_S$?

POODLE

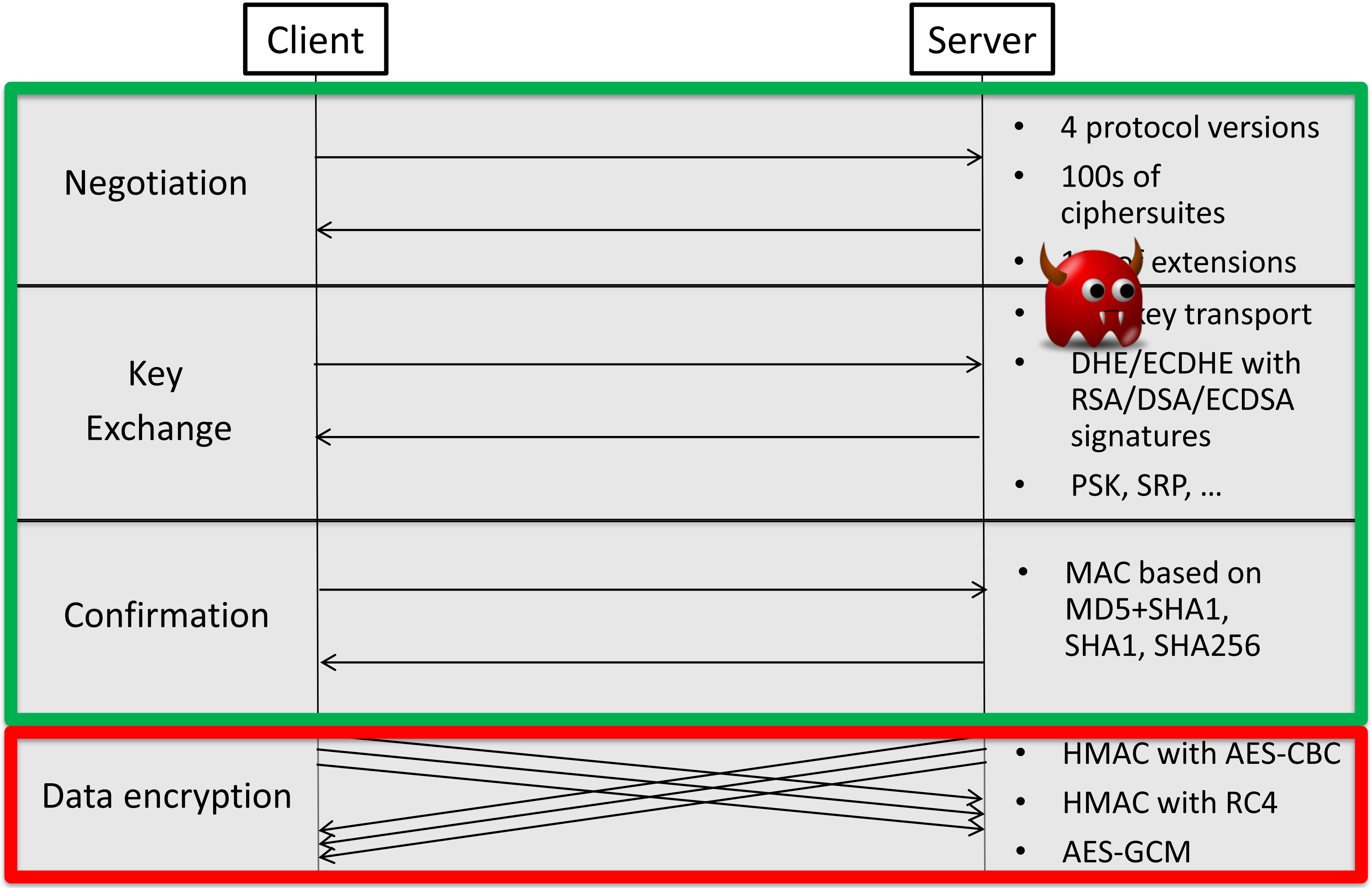
Client		Server
Negotiation	Hello Messages	• 4 protocol versions • 100s of ciphersuites • 10s of extensions
Key Exchange		• RSA key transport • DHE/ECDHE with RSA/DSA/ECDSA signatures • PSK, SRP, …
Confirmation	Finished Messages	• MAC based on MD5+SHA1, SHA1, SHA256
Data encryption		• HMAC with AES-CBC • HMAC with RC4 • AES-GCM

[Dowling and Stebila 2015]

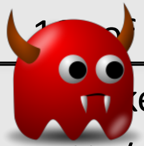
LOGJAM

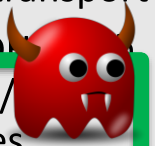

Client C
MitM
Server S
Knows sk_C, pk_S
$config_C$: G_{2048}, G_{512}
Knows sk_S, pk_C
$config_S$: G_{2048}, G_{512}
$[G_{2048}, G_{512}]$
$[G_{512}]$
$[G_{512}]$
$m_1 = g^x \; mod \; p_{512}$
$m_2 = g^y \; mod \; p_{512}$
$sign(sk_S, hash(m1||m2))$
$sign(sk_S, \text{transcript}')$?
k= $kdf(g^{xy} \; mod \; p_{512})$
$y = dlog(m_2)$
$k = kdf(g^{xy} \; mod \; p_{512})$
$k = kdf(g^{xy} \; mod \; p_{512})$
$mac(k, \text{transcript}')$
$mac(k, \text{transcript})$


BAD


Client
Server
Negotiation
4 protocol versions
100s of ciphersuites
extensions
Key Exchange
ey transport
DHE/ECDHE with RSA/DSA/ECDSA signatures
PSK, SRP, …
Confirmation
MAC based on MD5+SHA1, SHA1, SHA256
Data encryption
HMAC with AES-CBC
HMAC with RC4
AES-GCM

	Client	Server	
Negotiation	→ ←		• 4 protocol versions • 100s of ciphersuites • 1… of extensions
Key Exchange	→ ←		• …ey transport • DHE/ECD… • RSA/DSA/… signatures • PSK, SRP, …
Confirmation	→ ←		• MAC based on MD5+SHA1, SHA1, SHA256
Data encryption	⇄		• HMAC with AES-CBC • HMAC with RC4 • AES-GCM





$\mathbf{md5}(m_1 \,\|\, m'_2)$
$=\mathbf{md5}(m'_1 \,\|\, m_2)$

Downgrade secure configurations

- Downgrade protection (DP) only if
  - config_C requires good public keys and signatures scheme
  - config_S has preference for downgrade secure version

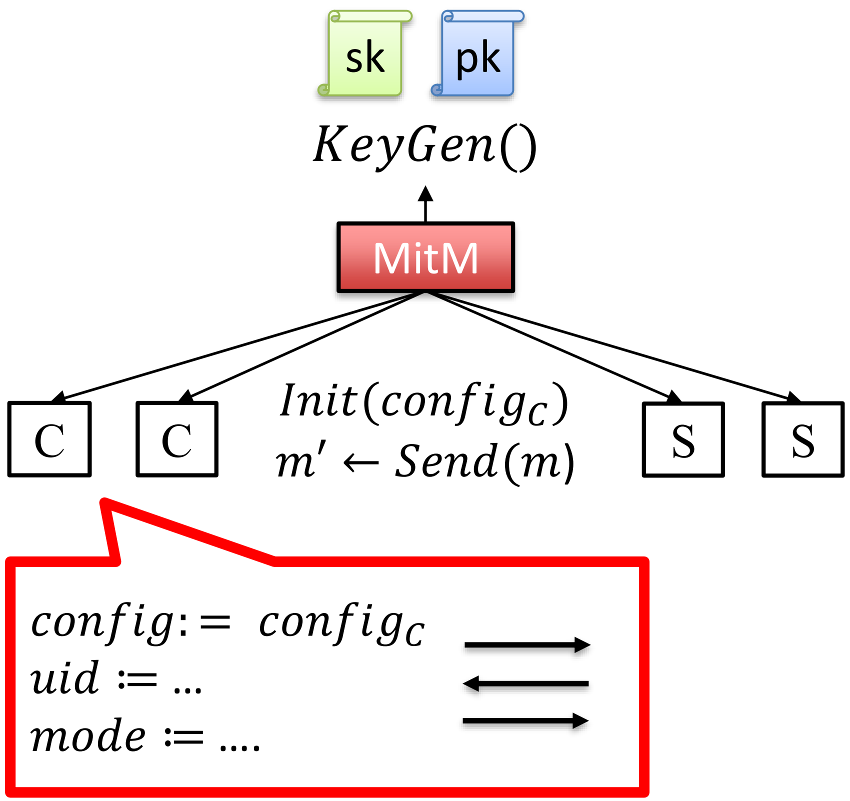
- Clients and servers interoperate with everyone;
  get desired mode **only** when DP($config_C$, $config_S$).

Protocol execution model

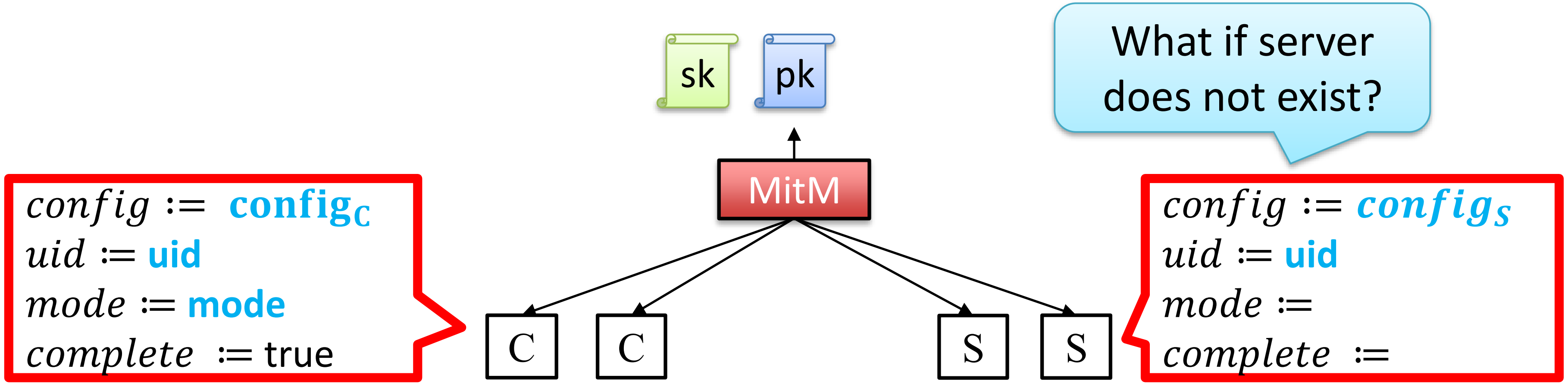
Adversary controls generation of keys and sessions

Configurations:
  algorithms and keys supported by sessions

Sessions assign **variables**



Downgrade security



DP($C.config, S.config$) but

$$mode \neq \text{Nego}(C.config, S.config)$$

Our contribution

1. Definition that tolerate weak algorithms
   - and capture downgrade attacks
2. Modular proof strategy

- Analyse downgrade security of SSH, IKE, ZRTP, **TLS**

- Prove downgrade security for SSH and **TLS 1.3**

NEW! New countermeasures designed together with core-design team of TLS 1.3

Reducing complex real-world protocol analysis …

Network Working Group
Internet-Draft
Obsoletes: 5077, 5246, 5746 (if approved)
Updates: 4492, 6066, 6961 (if approved)
Intended status: Standards Track
Expires: November 23, 2016

E. Rescorla
RTFM, Inc.
May 22, 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-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

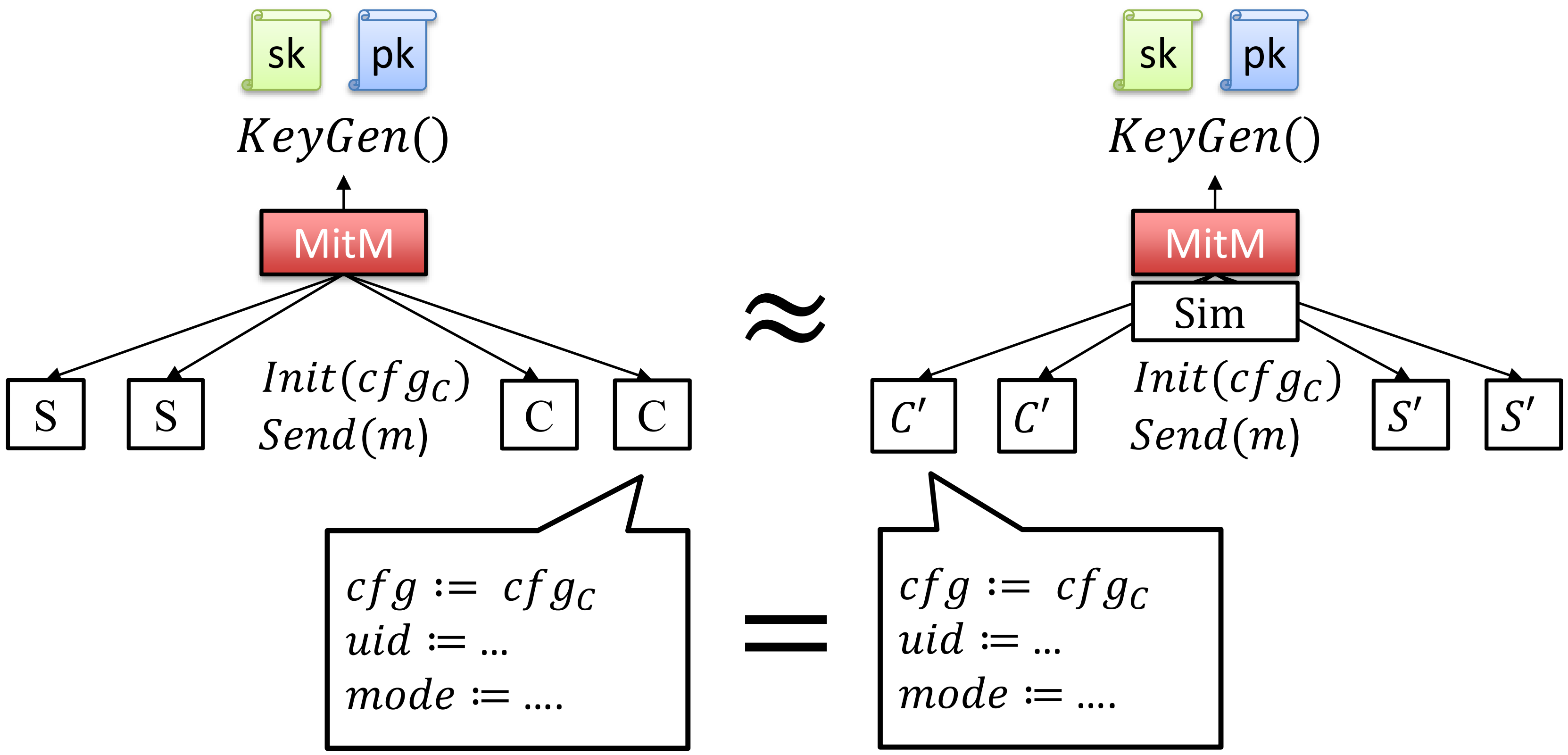
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 23, 2016.

Copyright Notice

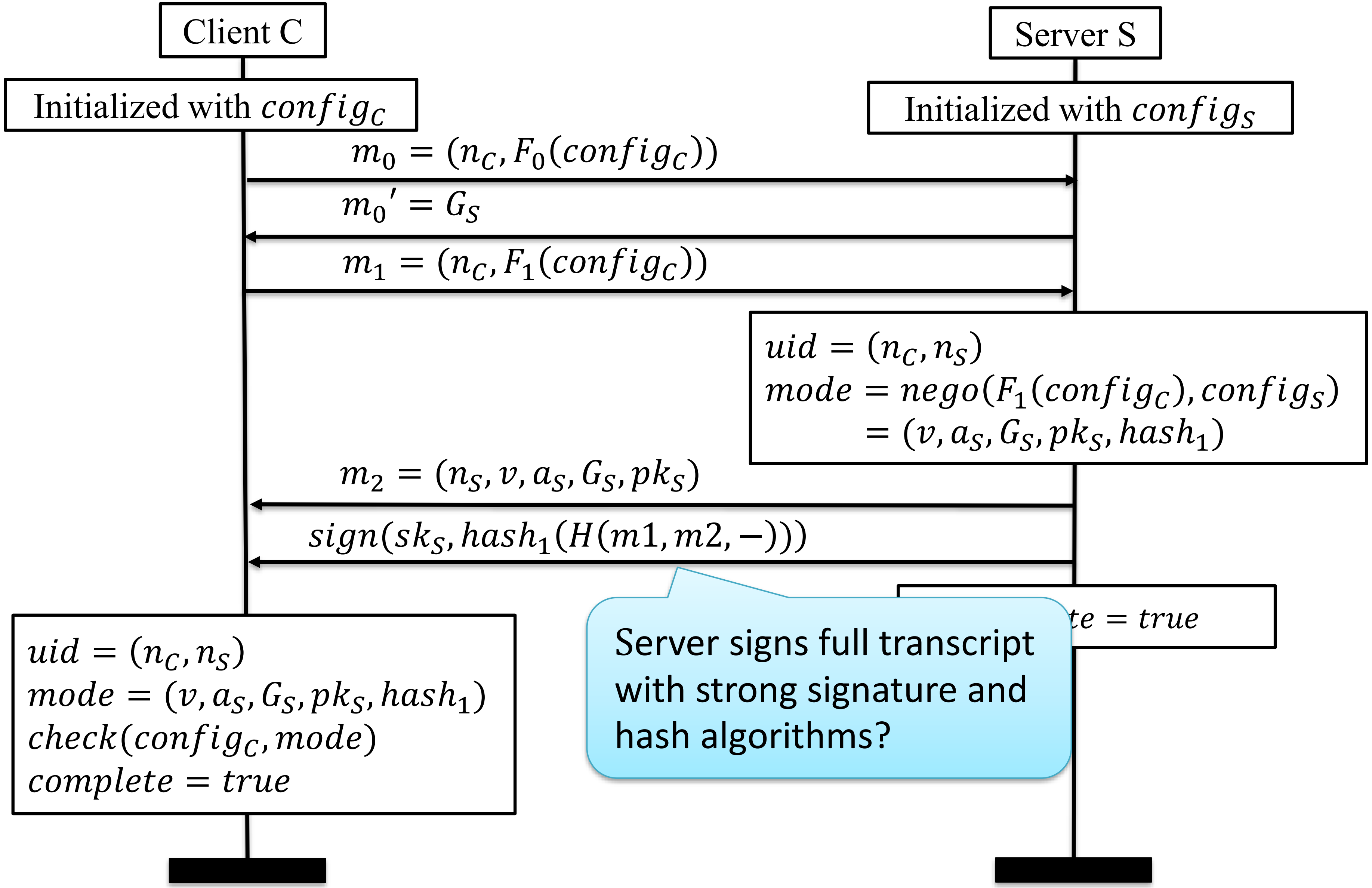
Table of Contents

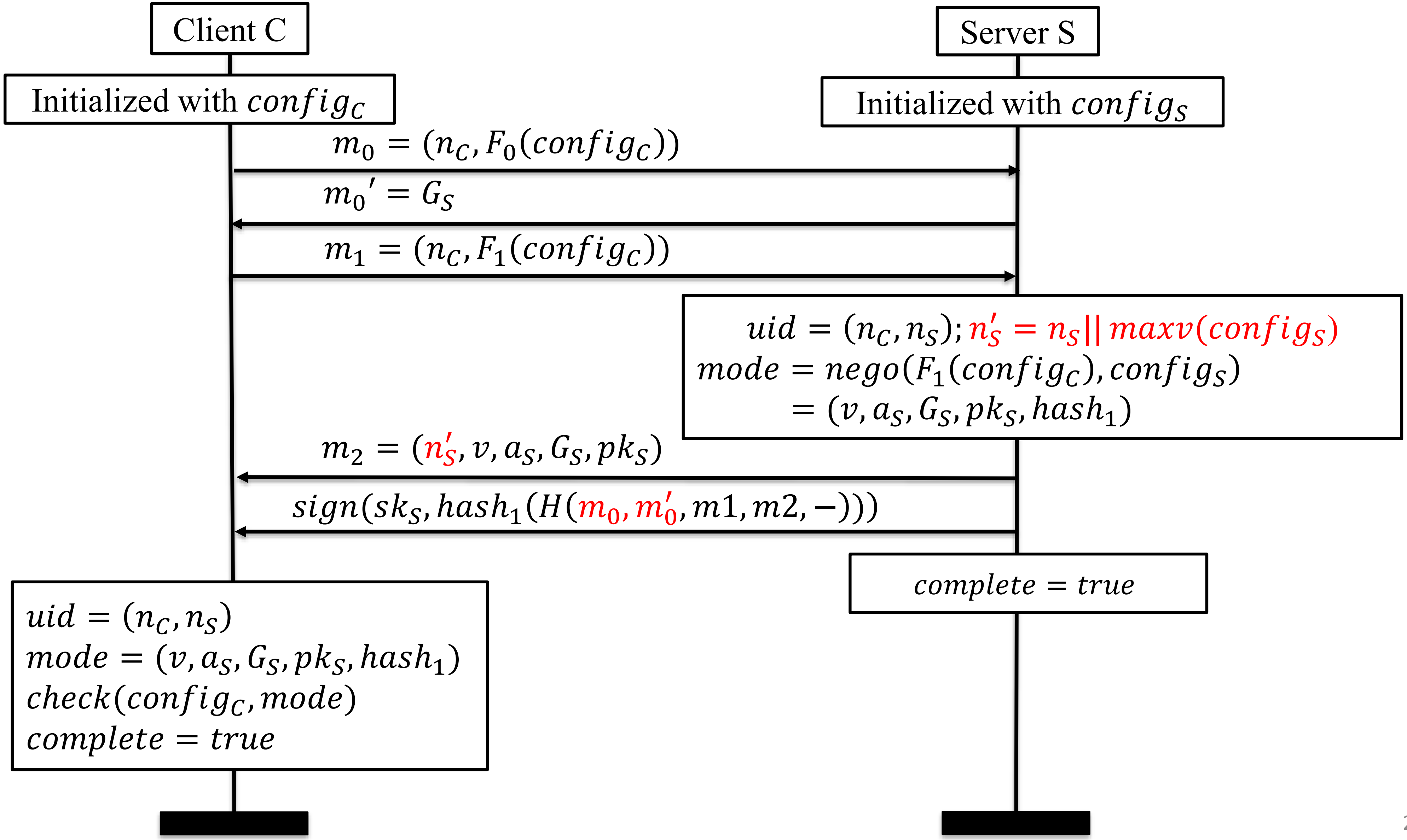
… using simulation …



[Rogaway and Steger 2009]

… into analysis of downgrade sub-protocol (TLS 1.3)




Client C
Server S
Initialized with $config_C$
Initialized with $config_S$
$m_0 = (n_C, F_0(config_C))$
$m_0{}' = G_S$
$m_1 = (n_C, F_1(config_C))$
$uid = (n_C, n_S); n'_S = n_S || maxv(config_S)$
$mode = nego(F_1(config_C), config_S)$
$= (v, a_S, G_S, pk_S, hash_1)$
$m_2 = (n'_S, v, a_S, G_S, pk_S)$
$sign(sk_S, hash_1(H(m_0, m'_0, m1, m2, -)))$
$complete = true$
$uid = (n_C, n_S)$
$mode = (v, a_S, G_S, pk_S, hash_1)$
$check(config_C, mode)$
$complete = true$

Downgrade security of TLS 1.3

- Good news:
  TLS 1.3 now has secure downgrade sub-protocol
  - **nonce and signatures**: unique server signs all network input to $nego$ and result.
- What do we do about version downgrade?
  - Can an attacker downgrade TLS 1.3 to TLS 1.2 and remount Logjam?

Version downgrade resilience

- TLS 1.3 server signatures cover versions
  But TLS 1.2 signatures **do not** cover the version

- How do we patch TLS 1.2 to prevent downgrades?
  - Finished messages cannot help
  - Look away: put max server version in server nonce
    signed in all versions of TLS

- Good news: DP($config_C, config_S$) for TLS 1.0-1.3 if
  - countermeasure implemented
  - no RSA key transport

Downgrade Resilience in Key Exchange

https://www.mitls.org/

https://eprint.iacr.org/2016/072