How (not) to use TLS between 3 parties

= Formal Model for **A**uthenticated and **C**onfidential **C**hannel **E**stablishment over **3** Parties=

---

presented by **Ioana Boureanu (Univ. of Surrey, SCCS, UK)**
given at the Open University, 24th of November 2016

a collaboration with K. Bhargavan (Inria, Paris, France)
C. Onete (IRISA, Rennes, France),
P. A. Fouque (UR1, Rennes, France),
B. Richard (Orange)

Based on a paper accepted for publication at Euro S&P 2017

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

Use of Insecure Channels....

Web (HTTP) traffic threat

- eavesdropping, impersonating, theft
  - Examples
    - Un-trusted ISPs
    - Vicious attackers
    - Mass surveillance …

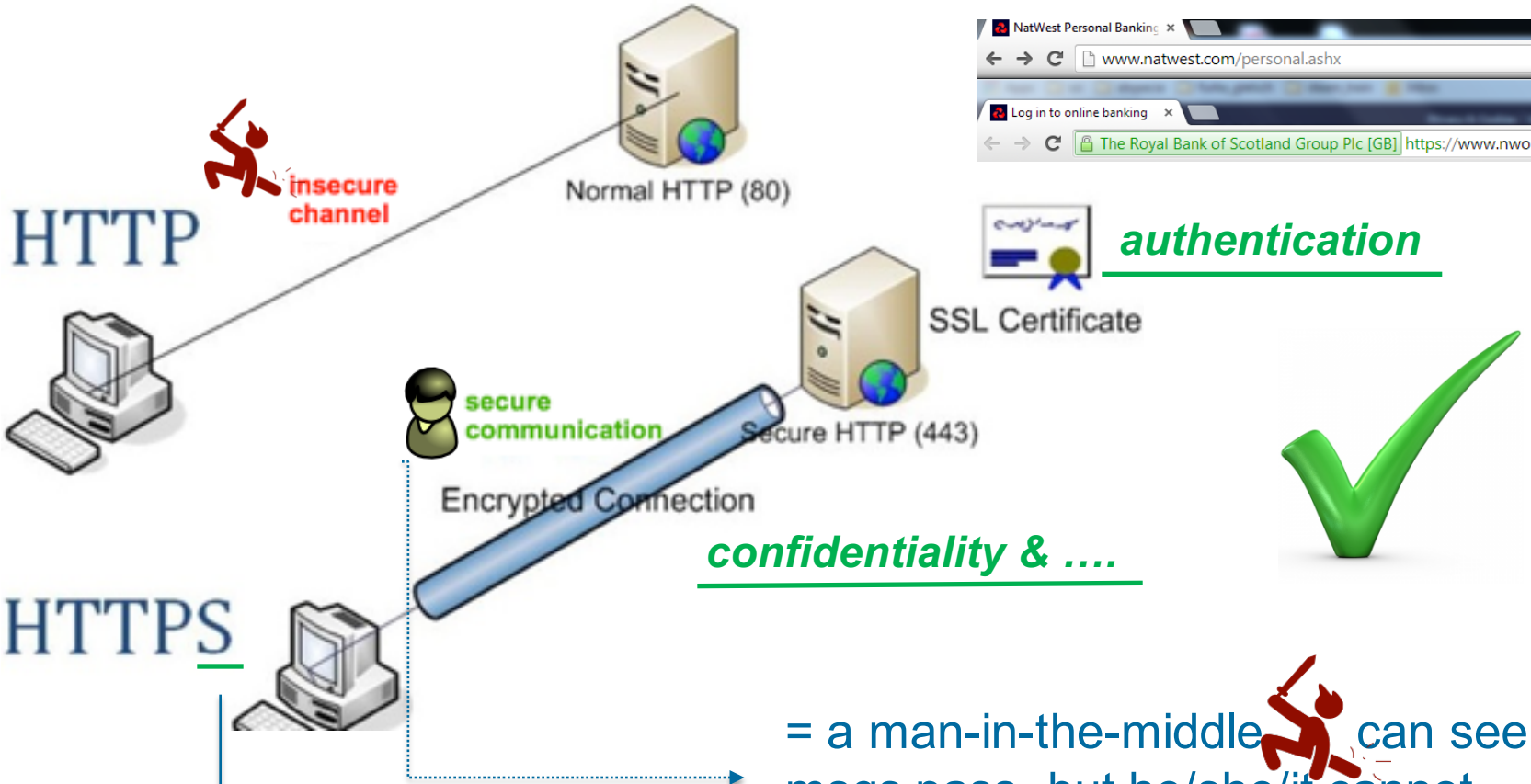
Mobile-networks threat (2G, 3G, 4G)

….

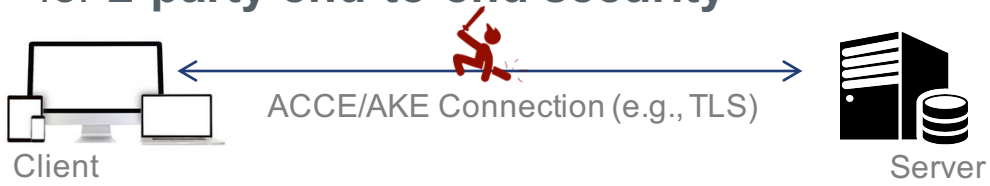
Scale of consumer cybercrime only (Norton by Symantec, 2013 report)

- 1 MILLION+ VICTIMS DAILY
- Costs: $ 298 per victim, 50% more than in 2012

Why SSL/TLS ? ... e.g., HTTP vs. HTTPS

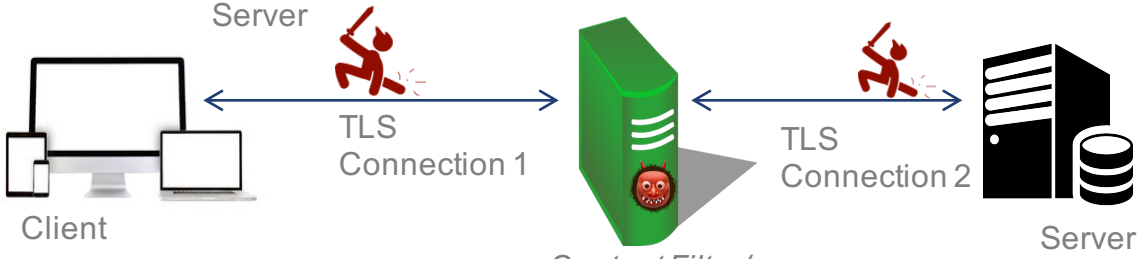


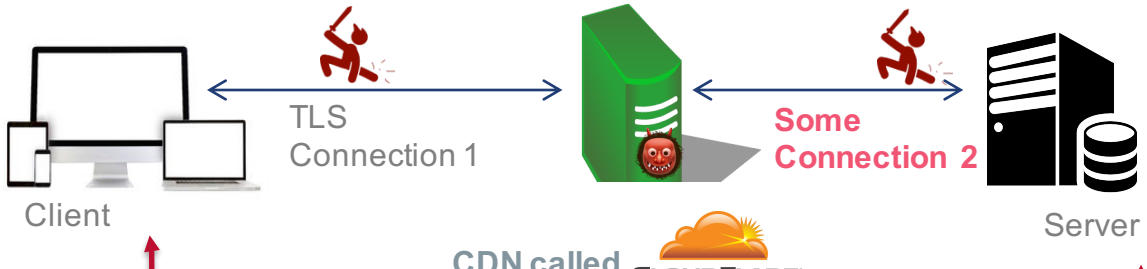
ACCE or AKE (authenticated key exchange) security models, and protocols implementing them (provably or otherwise) were conceived for **2-party end-to-end security**



!!

What guarantees do/should/could we have for 3-party ACCE??





!! product called **Keyless SSL** **!!**

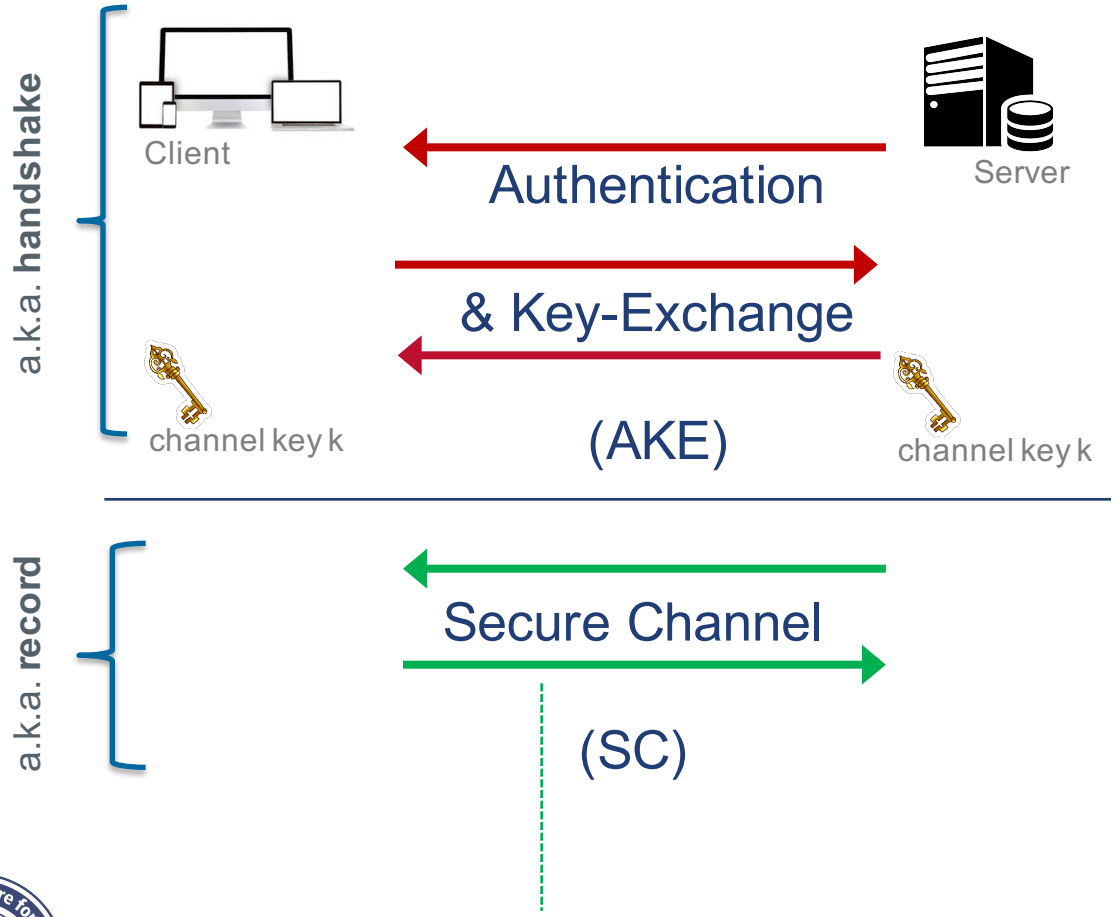
How (not) to use TLS between 3 parties

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

Towards Secure Communication over Insecure Channels

(Authenticated) Key-Exchange



Done using the crypto "toolbox":

Encryption:

Symmetric-key encryption:
- block ciphers, stream ciphers

Public-key encryption:
- RSA, ElGamal, …

Authentication:

MACs:
- shared-key between signer and verifier

Digital signatures:
- private key used for signing
- public key used for verification

(2-party) ACCE/AKE Threat-Model: Background

A man-in-the-middle adversary 😈; Aim: break channel-security

- 😈 can interact in multiple sessions, with several parties
- 😈 can corrupt parties, to learn long-term keys
- 😈 can reveal computed session keys

Authentication (mutual/unilateral)

Channel security

Extra guarantee:
forward-secrecy = if 😈 corrupts a user now and gets hold of its long-term secret key, then 😈 cannot break the security of past sessions

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

Preamble: The TLS/SSL Handshake (RSA-mode)

Client C

$\mathrm{sk}_S, \mathrm{pk}_S, \mathrm{Cert}_S$

Server S

N_C, ConfigList, ExtList →

← N_S, Config, Ext

← $\mathrm{KE}_S = (pk_S, \mathrm{Cert}_S)$

Choose **pmk**
Set: $\boldsymbol{msk} = \mathrm{HMAC}_{pmk}(N_C, N_S)$
$KE_C = \mathrm{Enc}_{pk_S}(\mathrm{pmk})$
$(K_C, K_S) = \mathrm{HMAC}_{msk}(N_C, N_S)$
$\mathrm{CFin} = \mathrm{HMAC}_{msk}(H[N_C \ldots KE_C])$

$\mathrm{KE}_C, \{\mathrm{CFin}\}_{K_C}$ →

Decrypt KE_C to get pmk
Compute msk, (K_C, K_S)
Decrypt $\{\mathrm{CFin}\}_{K_C}$ and check $CFin$
$\mathrm{SFin} = \mathrm{HMAC}_{msk}(H[N_C \ldots \mathrm{CFin}])$

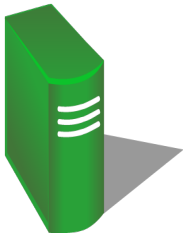
← $\{\mathrm{SFin}\}_{K_S}$

- TLS 1.2 in specific modes with specific primitives/assumptions → provably secure
- session resumption: shortcutting the hanshake, from old **msk**, no sk_S being used at the current run

Preamble: Classical TLS over CDNs vs. Cloudflare's Keyless SSL

Client

Classical CDN, Edge-Server



$sk_S^{Impersonation} = skS$

$pk_S^{Impersonation} = pkS$

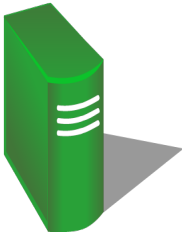
$Cert_S^{Impersonation} = CertS$

Origin Server **S**

$sk_S^{Origin}, pkS^{Origin},$
$Cert_S^{Origin}$

Client

Cloudflare's Keyless Edge-ServerCDN



$pk_S^{Impersonation} = pkS$

$Cert_S^{Impersonation} = CertS$

Origin Server **S**

$sk_S^{Origin}, pkS^{Origin},$
$Cert_S^{Origin}$

$sk_S, pkS, Cert_S$

Cloudflare's Keyless SSL

Client C

pk_S, $Cert_S$

Keyless SSL Edge-Server MW

sk_S, pk_S, $Cert_S$

Origin Server S

N_C, ConfigList, ExtList

N_S, Config, Ext

$KE_S = (pk_S, Cert_S)$

Choose pmk

Set: $msk = HMAC_{pmk}(N_C, N_S)$

$KE_C = Enc_{pk_S}(pmk)$

$(K_C, K_S) = HMAC_{msk}(N_C, N_S)$

$CFin = HMAC_{msk}(H[N_C \dots KE_C])$

KE_C, $\{CFin\}_{K_C}$

KE_C

pmk

! uses his sk to produce a **channel key for C—MW, to which he has no access**

Get **msk**, (K_C, K_S), check CFin

$SFin = HMAC_{msk}(H[N_C \dots CFin])$

$\{SFin\}_{K_S}$

! gets **pmk**

Surrey Centre for Cyber Security SCCS

Keyless SSL's Pb1: Lack of Accountability

N_C, ConfigList, ExtList

...

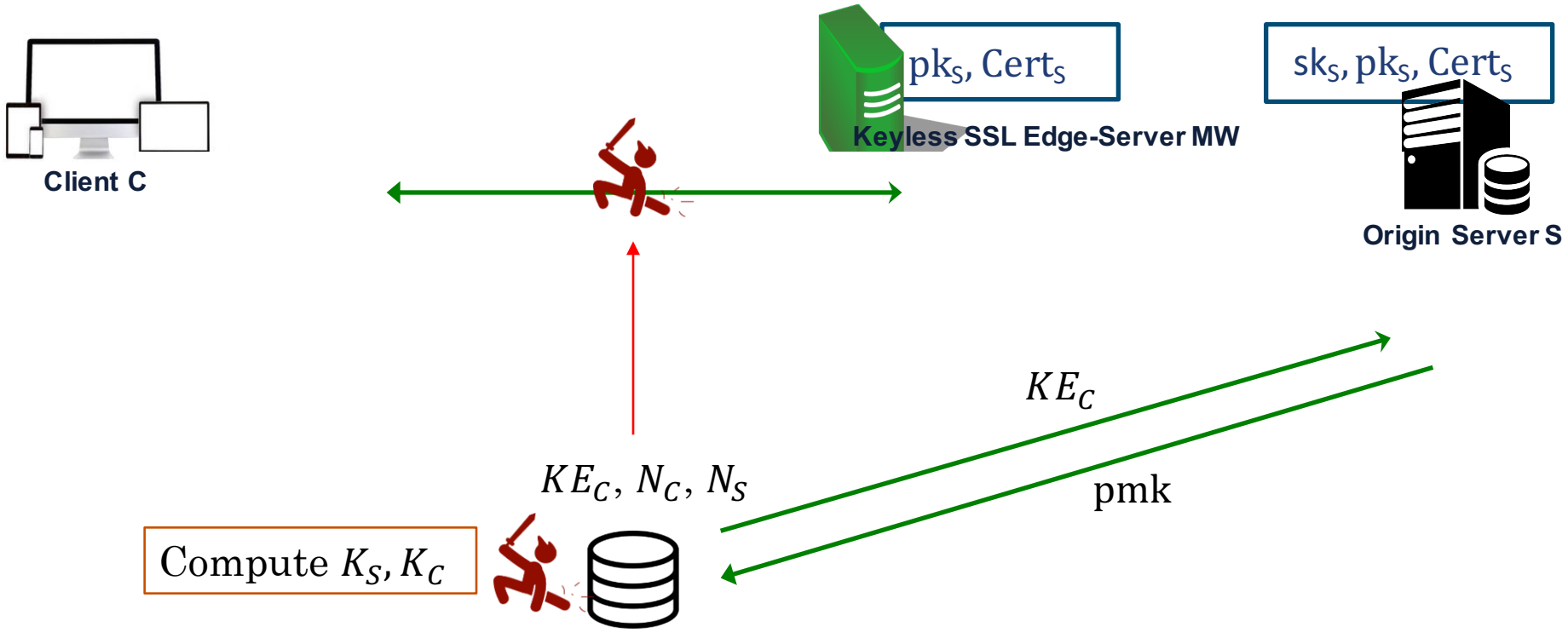
$\{SFin\}_{K_S}$

Malware

The CDN can send what it pleases *on behalf of S*, without S's knowledge/control!

If the server S knew the channel-key, the S could monitor/audit the CDN's behaviour!

Keyless SSL's Pb2: One, All-Compromising Malicious CDN



A malicious CDN can compromise all sessions! Mass surveillance?!

If the server S got more than just a out-of-context KE_C (e.g., C-side session-data), then this attack would be harder for malicious CDNs

How (not) to use TLS between 3 parties

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

Keyless SSL's Main Pbs: Recall

Client C

$pk_S, Cert_S$

Keyless SSL Edge-Server MW

$sk_S, pk_S, Cert_S$

Origin Server S

N_C, ConfigList, ExtList

N_S, Config, Ext

$KE_S = (pk_S, Cert_S)$

Choose pmk

Set: $msk = HMAC_{pmk}(N_C, N_S)$

$KE_C = Enc_{pk_S}(pmk)$

$(K_C, K_S) = HMAC_{msk}(N_C, N_S)$

$CFin = HMAC_{msk}(H[N_C \dots KE_C])$

$KE_C, \{CFin\}_{K_C}$

Get **msk**, (K_C, K_S), check CFin

$SFin = HMAC_{msk}(H[N_C \dots CFin])$

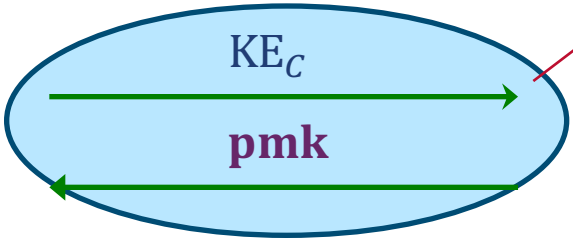
$\{SFin\}_{K_S}$

No C-session data to S =>

- channel security pb's by malicious MW
- no audit possible by S

pmk/msk to MW:

- session resumption by MW



Keyless SSL's Fix: First-Attempt

Client C

$pk_S, Cert_S$

Keyless SSL Edge-Server MW

$sk_S, pk_S, Cert_S$

Origin Server S

N_C, ConfigList, ExtList

N_S, Config, Ext

$KE_S = (pk_S, Cert_S)$

Choose pmk

Set: $msk = HMAC_{pmk}(N_C, N_S)$

$KE_C = Enc_{pk_S}(pmk)$

$(K_C, K_S) = HMAC_{msk}(N_C, N_S)$

$CFin = HMAC_{msk}(H[N_C \dots KE_C])$

$KE_C, \{CFin\}_{K_C}$

$[N_C, \dots, \{CFin\}_{K_C}]$

Check validity

pmk

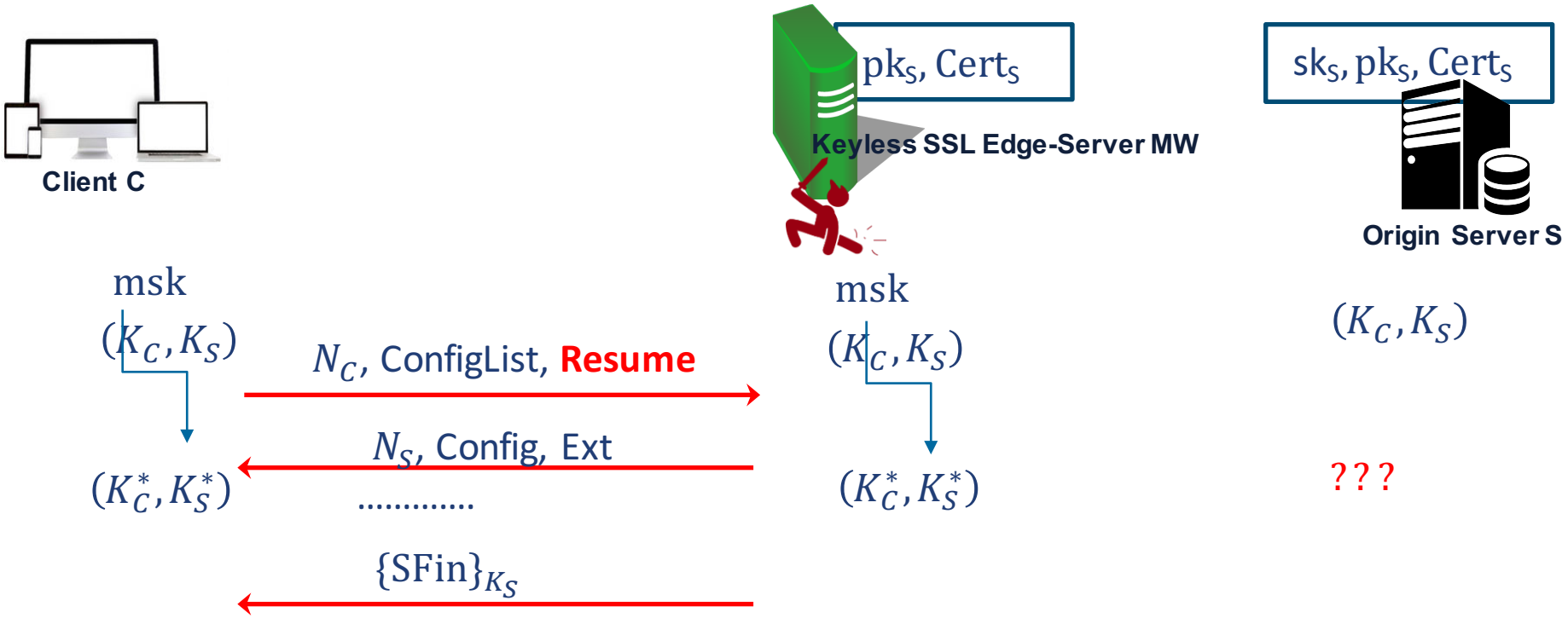
Get msk, (K_C, K_S), check CFin

$SFin = HMAC_{msk}(H[N_C \dots CFin])$

$\{SFin\}_{K_S}$

Surrey Centre for SCCS Cyber Security

Keyless SSL's Fix-Attempt: Resumption vs Accountability



Resumption: run a shorter handshake => computation of session-keys related to a previous, full handshake

Given msk, MW just needs a new tuple of nonces for new session-keys

Accountability??

Keyless SSL's Fix: Second Attempt

Client C

$pk_S, Cert_S$

Keyless SSL Edge-Server MW

$sk_S, pk_S, Cert_S$

Origin Server S

N_C, ConfigList, ExtList

N_S, Config, Ext

$KE_S = (pk_S, Cert_S)$

Choose pmk

Set: $msk = \text{HMAC}_{pmk}(N_C, N_S)$

$KE_C = \text{Enc}_{pk_S}(\text{pmk})$

$(K_C, K_S) = \text{HMAC}_{msk}(N_C, N_S)$

$\text{CFin} = \text{HMAC}_{msk}(H[N_C \ldots KE_C])$

MW chooses all parameters

Cross-protocol, Auth. attacks

$KE_C, \{\text{CFin}\}_{K_C}$

$[N_C, \ldots, \{\text{CFin}\}_{K_C}]$

(K_C, K_S, SFin)

Check validity

$\{\text{SFin}\}_{K_S}$

Fixed Keyless TLS 1.2

Client C

$pk_S, Cert_S$

Keyless SSL Edge-Server MW

$sk_S, pk_S, Cert_S$

Origin Server S

N_C, ConfigList, ExtList

N_S, Config, Ext

$KE_S = (pk_S, Cert_S)$

Request

N_S

Choose pmk

Set: $msk = \text{HMAC}_{pmk}(N_C, N_S)$

$KE_C = \text{Enc}_{pk_S}(\text{pmk})$

$(K_C, K_S) = \text{HMAC}_{msk}(N_C, N_S)$

$\text{CFin} = \text{HMAC}_{msk}(H[N_C \ldots KE_C])$

Unavoidable

S runs entire protocol!

$\text{KE}_C, \{\text{CFin}\}_{K_C}$

$[N_C, \ldots, \{\text{CFin}\}_{K_C}]$

Check validity

(K_C, K_S, SFin)

$\{\text{SFin}\}_{K_S}$

Fixed Keyless TLS 1.2 (RSA Mode) -- Notes

Authentication and Secure Channel:

- N_S generated honestly
- KE_C given in full context, allows S (honest) to prevent channel-security attacks by MW corruption

Accountability (by S for the C—MW link):

- MW forwards the nonces of MW and C, encrypted CFin, and KE_C
  - S can verify the forwarded nonces are correct as per Cfin and KE_C
- S sends directly the session keys and encrypted SFin
  - No session resumption

Other features:

- Security w.r.t. to new formal ACCE model
- More security guarantees, under some assumptions, such as *content soundness* (i.e., "MW can only deliver contracted material")

Fixing Keyless SSL – Some More Results & Discussions

Original Keyless TLS 1.2. in DHE mode

-- It exhibits cross-protocol attack

-- It ensures no accountability & no content soundness

-- Unfortunately, our Fixed Keyless TLS 1.2 in DHE mode has the same drawbacks as our Fixed Keyless TLS 1.2 in RSA mode (i.e., large PKI, heavy server-side computation)

Keyless TLS 1.3

-- It did not exist in CloudFlare's original proposal

-- In our paper, we propose a Keyless TLS 1.3 which

- does not allow resumption
- is more efficient than Fixed Keyless TLS 1.2 and
- needs a lighter PKI

A General Tradeoff:

-- accountability vs limited resumption

-- in the full version, we show how to attain 3(S)ACCE -security and allow session resumption, if we allow the client to be aware that the handshake is legitimately proxied

How (not) to use TLS between 3 parties

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

The "Proxied AKE" Infrastructure -> Our 3ACCE Model

Three-party system:

- Client, Server, Middleware (MW)
- Server owns contents $\omega_1, \ldots \omega_n$
  - Each ω_i associated with $(sk_i^S, pk_i^S, \mathrm{Cert}_i^S)$
- MW agrees on contract with Server that:
  - MW can later cache/"process" ω_i and serve it to clients
  - MW has its own credentials (sk_{MW}, pk_{MW})
  - For each contracted ω_i, MW gets $(pk_i^{S,MW}, \mathrm{Cert}_i^{S,MW})$, *maybe* $sk_i^{S,MW}$

Client C

Middleware MW

sk_{MW}, pk_{MW}

$\omega_j, pk_j^{S,MW}, \mathrm{Cert}_j^{S,MW}$

$sk_j^{S,MW}$

Origin Server S

$\omega_1, \ldots \omega_n$

$sk_i^S, pk_i^S, \mathrm{Cert}_i^S \quad \forall i$

Secure "Proxied AKE" or the 3ACCE Security

Composition of two 2-party channels (C-MW & MW-S)

- C-MW is always unilaterally authenticated
- MW-S is always mutually authenticated
- C-S (direct) is always unilaterally authenticated

We defined four security notions

- Authentication
- Channel security

} Adapted from the 2-party case

- Accountability: if MW impersonates S, then S knows key
- Content soundness: MW cannot deliver uncontracted content

Main technical difficulty: session partnering

Security Definitions' Crux: Session Partenering

Protocol is executed by parties

- Each execution is a party instance
- Party instances execute protocol sessions, which have sid's
- Each party instance keeps track of:
  - Session ID sid – e.g. randomness and values used in key-computation
  - Partner ID pid – party with which one thinks they communicate

For CDN, pid could be server, while "real" partner is MW

  - Computed session key set K
  - Some other AKE technicalities (e.g., reveal bit, channel bit, etc.)

2-Partnering: 2 instances are partnered if they share sid's

But importantly...
partnering defines which sessions can be secured

There are two cases:

- Client is aware of MW (essentially 2-partnering)
- Client is unaware of MW (CDN/Keyless SSL)

If client is unaware of MW, there are two sub-cases:

- If MW needs S (Keyless SSL): four instances partnered:
  - Client instance, MW1 instance, MW2 instance, S instance
- If MW is autonomous (like in CDN): 2 instances partnered:
  - Client + MW1 instance
  - Partnering extends on 3 parties (and corrupting S is treated within)

Using 3-partnering this way allows us to re-use 2-party security definitions for authentication + channel security

New 3ACCE Guarantees: Accountability & Content Soundness

Accountability

- CDNs allow MW to impersonate S, with S's accord
- It is in S's interest not to care beyond that
- However, client has no way of distinguishing MW & S
- Solution:
  - Either make client aware of the MW
  - Or make sure MW unable to "hurt" client (by auditing secure channel)

Content Soundness

- MW only allowed to know some contents (by contact)
- Later, MW will contact S and ask to cache contents
- S must make sure only allowed contents are sent

Fixed Keyless: Provable Security

Theorem 1. Let Π be the 3(S)ACCE-K-SSL variant. We denote by P be the unilaterally-authenticated TLS 1.2 handshake, by P', the mutually-authenticated TLS 1.2 handhsake, and by Ψ, the transformation of P' to an AKE protocol by the computation of the export key ek. If the following conditions hold:

- If P is a 2-SACCE-secure protocol, P' is a 2-ACCE-protocol, and $\Psi(P')$ yields pseudorandom keys;
- For TLS-DHE: if the hash function H is collision-resistant and the signature scheme used to generate PSign is unforgeable;
- For TLS-RSA: if P guarantees channel security;

Then Π guarantees 3(S)ACCE-security[6].

How (not) to use TLS between 3 parties

PRESENTATION OUTLINE

1. Context/Aim: achieve secure communication over insecure channels
2. Authenticated and Confidential Channel Establishment (ACCE) Model -- 2 parties
3. Attacks on Cloudflare's Keyless SSL: How 2ACCE-security breaks with 3 parties
4. A Provably Secure Keyless SSL Variant
5. ACCE with 3 parties (3ACCE): A Glimpse of the Formalisation
6. Food for Thought

Keyless SSL and Other TLS "Compositions" over 3 Parties

UNIVERSITY OF SURREY

CDNs, Filtering, Proxy-ing

- Uses a "cache/process then deliver" strategy to improve efficiency in content delivery over HTTPS:// or to filter content. etc.
- Provide such services transparently to clients
- A single CDN/Proxy can serve many content owners simultaneously

- TLS/SSL was not designed to be run in 2+, i.e., be composably secure
- These services were not designed with client-security and privacy in mind
- Bespoke solutions like Keyless SSL can be even worse than simple "TLS sequencing"
- These services can provide a mass surveillance tool since, e.g., a lot of information passes through a single CDN!
- These services do not allow clients to make informed decisions based on whether they communicate with a CDN/proxy, etc. or the server directly

Surrey Centre for SCCS Cyber Security

Maybe, the client should know and would choose security over efficiency!

THANK YOU!

QUESTIONS?

- The speaker thanks C. Onete for some of the material/illustrations on these slides.
- The copyright of certain figures/icons/pictures used within is not attributed to the presenter. These were employed for illustration purposes only and not to be re-used without further research into their copyright.