lez 24, 26/11

TLS 1.3 at a glance
&
Perfect Forward Secrecy

1.2 in 2008, molto diversi!

TLS 1.3: Approved in 2018

Protocol Action: 'The Tran

Sicuro | https://www.ietf.org/mail-archive/web/ietf-announce/current/msg17592.html

App | The Future of Netwo | Governor of Poker 3 | BTC-Tech | Class Prof | HPSR2018 - Google | Windsurf. Mercatino | Stazione meteo Mar | Altri Preferiti

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

Protocol Action: 'The Transport Layer Security (TLS) Protocol Version 1.3' to Proposed Standard (draft-ietf-tls-tls13-28.txt)

- *To*: "IETF-Announce" <ietf-announce at ietf.org>
- *Subject*: Protocol Action: 'The Transport Layer Security (TLS) Protocol Version 1.3' to Proposed Standard (draft-ietf-tls-tls13-28.txt)
- *From*: The IESG <iesg-secretary at ietf.org>
- *Date*: Wed, 21 Mar 2018 03:02:02 -0700
- *Archived-at*: <https://mailarchive.ietf.org/arch/msg/ietf-announce/IhM9JJHVs_ZeK-_1eaVZrqxbnL8>
- *Auto-submitted*: auto-generated
- *Cc*: The IESG <iesg at ietf.org>, draft-ietf-tls-tls13 at ietf.org, Sean Turner <sean at sn3rd.com>, Kathleen.Moriarty.ietf at gmail.com, tls-chairs at ietf.org, rfc-editor at rfc-editor.org, sean at sn3rd.com, tls at ietf.org
- *Delivered-to*: ietf-announce at ietfa.amsl.com
- *List-archive*: <https://mailarchive.ietf.org/arch/browse/ietf-announce/>
- *List-help*: <mailto:ietf-announce-request@ietf.org?subject=help>
- *List-id*: "IETF announcement list. No discussions." <ietf-announce.ietf.org>
- *List-post*: <mailto:ietf-announce@ietf.org>
- *List-subscribe*: <https://www.ietf.org/mailman/listinfo/ietf-announce>, <mailto:ietf-announce-request@ietf.org?subject=subscribe>
- *List-unsubscribe*: <https://www.ietf.org/mailman/options/ietf-announce>, <mailto:ietf-announce-request@ietf.org?subject=unsubscribe>

```
The IESG has approved the following document:
- 'The Transport Layer Security (TLS) Protocol Version 1.3'
  (draft-ietf-tls-tls13-28.txt) as Proposed Standard

This document is the product of the Transport Layer Security Working Group.

The IESG contact persons are Kathleen Moriarty and Eric Rescorla.
```

Scrivi qui per eseguire la ricerca

11:25
01/05/2018

TLS 1.3: Approved in 2018

[Docs] [txt|pdf] [draft-ietf-tls-...] [Tracker] [Diff1] [Diff2] [IPR] [Errata]

PROPOSED STANDARD
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 8446
Obsoletes: 5077, 5246, 6961
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

E. Rescorla
Mozilla
August 2018

The Transport Layer Security (TLS) Protocol Version 1.3

Abstract

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

This document updates RFCs 5705 and 6066, and obsoletes RFCs 5077, 5246, and 6961. This document also specifies new requirements for TLS 1.2 implementations.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Fino a 1.2 c'erano varie criticità!

TLS 1.3: weak ciphers' pruning

(rimossi)

➔ **Pruned all symmetric algorithms considered legacy.**

- ⇨ SHA-1, MD5
- ⇨ RC4, DES, 3DES
- ⇨ AES-CBC
- ⇨ (plus a few more technical removals)

➔ **What is left?**

- ⇨ ALL (!) remaining ciphers are AEAD (Authenticated Encryption with Associated Data)
  - →AEAD: no more CCA (Padding Oracle, Lucky 13. etc).
- ⇨ Notation: TLS_AEAD_HASH (hash used for HKDF):
  - →TLS_AES_128_GCM_SHA256 Key derivation, encryption & integrity alg (authenticated cypher)
  - →TLS_AES_256_GCM_SHA384
  - →TLS_CHACHA20_POLY1305_SHA256
  - →TLS_AES_128_CCM_SHA256
  - → TLS_AES_128_CCM_8_SHA256

TLS 1.3 handshake

➔**TLS1.2: four pubkey handshake methods supported**

- ⇨**RSA** key transport (most common)
- ⇨**Anonymous** Diffie-Hellman (no auth. end-point, soffre MITM)
- ⇨**Fixed** Diffie-Hellman (certifica il public term coefficient)
- ⇨Diffie-Hellman **Ephemeral**

PRIMA, tra client/server hello negoziavo, oltre al cipher, ma anche come creare il segreto. Ora siamo "forzati" ad usare D.H. (flessibile al suo interno, es. elliptic curve)

➔**TLS1.3: removed all, except one!**

- ⇨Diffie-Hellman **Ephemeral** (uso SOLO key agreement)

➔**Why? Forward Secrecy!**

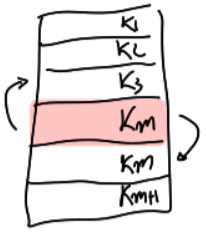
- ⇨What's this? See next slides ☺
- ⇨But also get rid of RSA issues (Bleichenbacker Oracle, …)

What is perfect forward secrecy?

➔**At the very minimum:**

⇨If a session key is compromised this should NOT compromise data exchanged neither BEFORE nor AFTER.

TLS non lo faceva già?



➔**Actually more than this!**

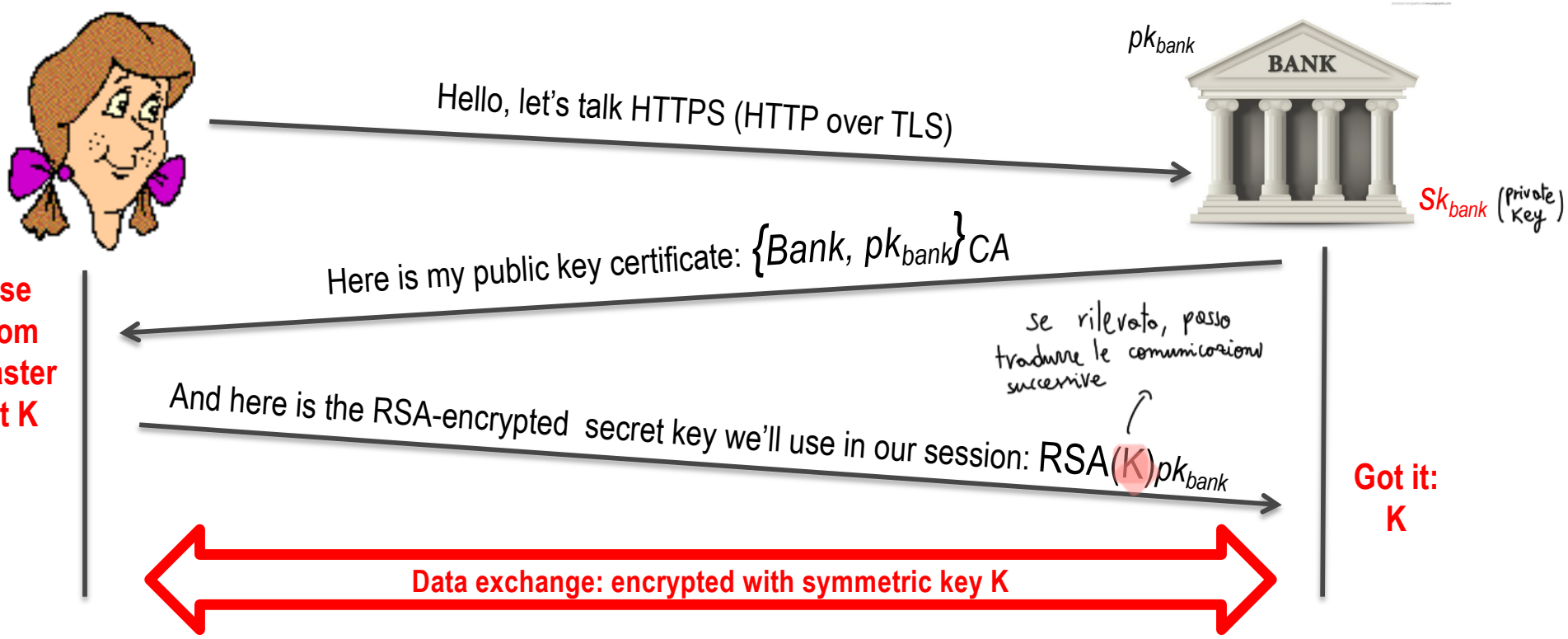
⇨if a long-term private key (e.g. of the server!!) is compromised at some time, this should also NOT affect data delivered BEFORE such time!

➔**In essence: PFS guarantees that session keys will not be compromised even if long-term secrets used in the session key exchange are compromised**

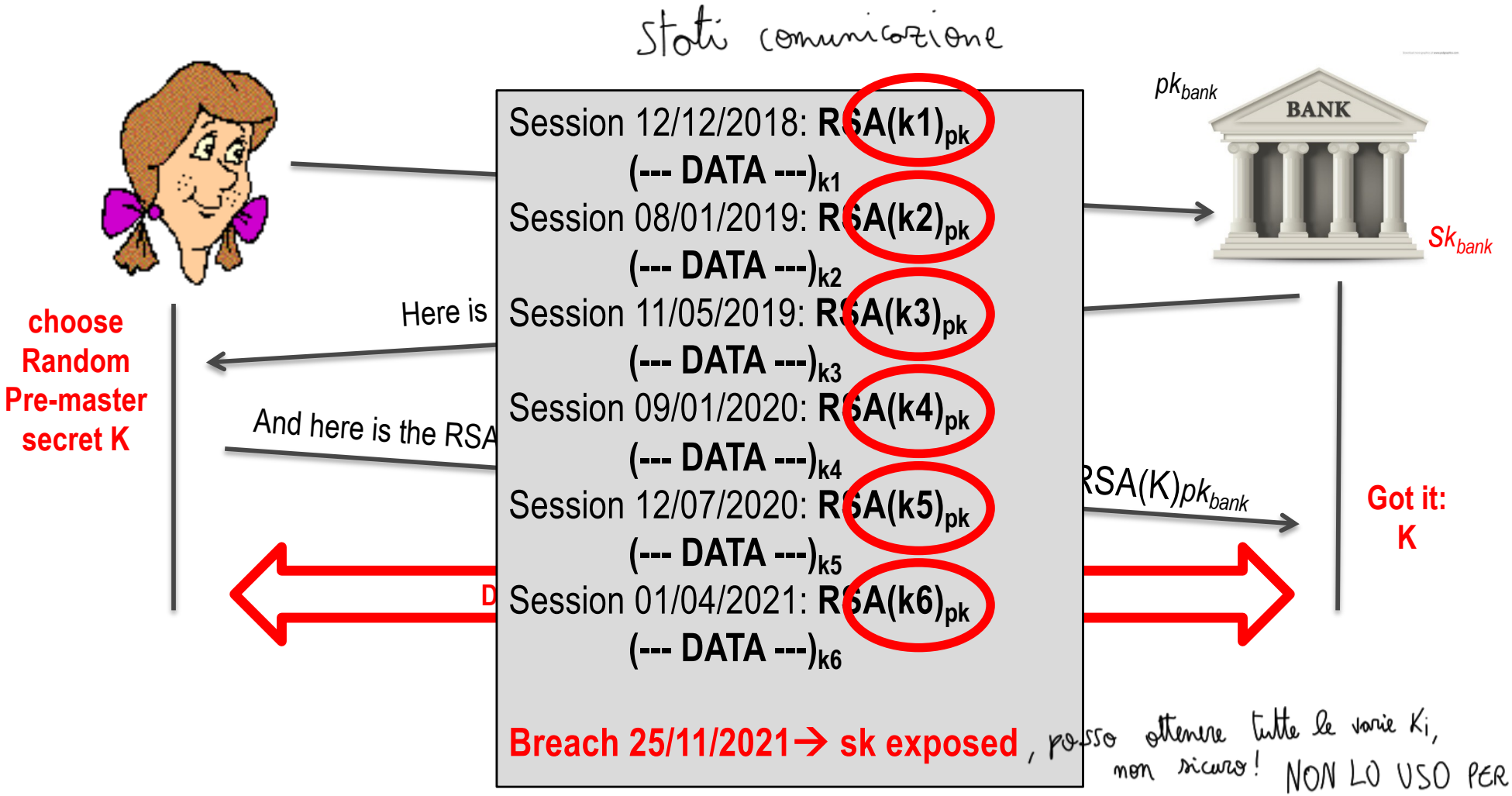
Why PFS is today more and more important?

- **Protects agains a more powerful attacker model**
  - Capable of **logging huge amount of data**
    - E.g. a mass-surveillance program capturing all network traffic (remember Snowden…)
  - Capable to **occasionally break a public/private key pair**
    - E.g. via a SW bug in the implementation – (remember heartbleed)

Traditional RSA key transport: no PFS!!



Traditional RSA key transport: no PFS!!



Fixed DH: even worse!!

Fixed
x, g^x

Fixed
y, g^y

$N1_{client}$, g^x

$N1_{server}$, g^y

Data exchange: K1 = HKDF(N1c, N1s, g^{xy})

nonces

• trovare K_2 non ci dà info su K_1

$N2_{client}$, g^x

$N2_{server}$, g^y

Data exchange: K2 = HKDF(N2c, N2s, g^{xy})

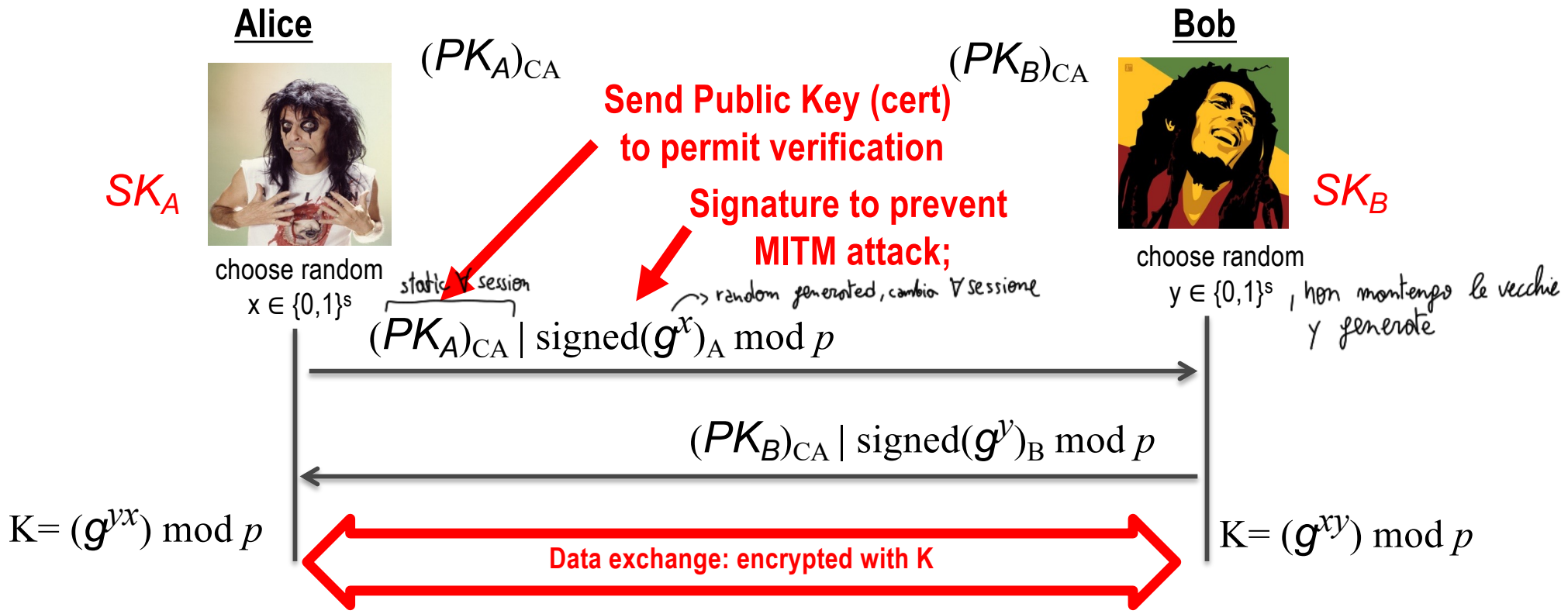
costanti

Breach 25/11/2021 → x (or y) exposed → g^{xy} disclosed!

→ compute K2 (nonces in hello msg) → And all past K!!

Ephemeral Diffie-Hellman: PFS!

- Ho due segreti, associato a signature $(PK_A)_{CA}$



If SK breaks, (ephemeral) session keys are not revealed! Forward Secrecy!

Amd even AFTER breach, if attacker if only passive (but MITM now possible)

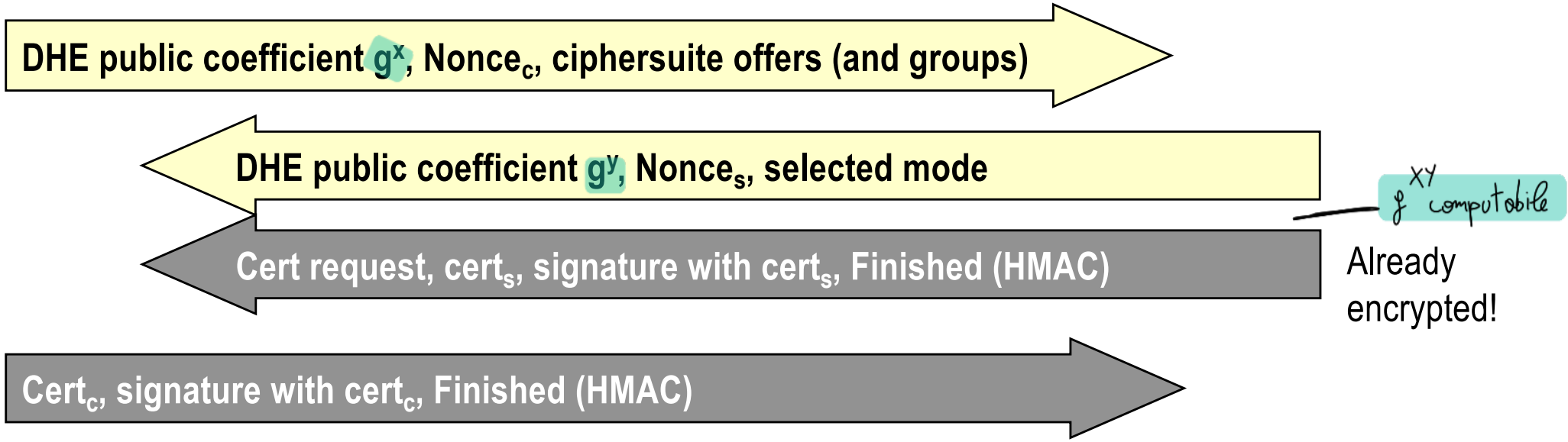
- Se avessi SK_B, non posso tornare indietro (trovare g^{xy} è inutile, buona solo per "sign"). SK_B serve per MITM.
- nel futuro posso? si, ma con MITM, cioè dopo breach e presa private key, DOPO posso fare MITM (active attack)

TLS 1.3 handshake:
Faster and more secure!

è semplificabile, avevo 4 way handshake per scegliere RSA, D.H., ... Ora posso mandare 'g^x' al 1° msg, e fare encrypt dal 2° msg.

➔With only DHE left (no more RSA transport):

- ⇨3-way handshake versus 4-way! → faster setup
- ⇨Perfect forward secrecy, migliora privacy poiché gran parte dell'handshake è protetta.



Side effect of early encryption: IDENTITY PROTECTION!
(certificates never transmitted in clear)
(not very useful in web, but keep in mind for different TLS scenarios…)

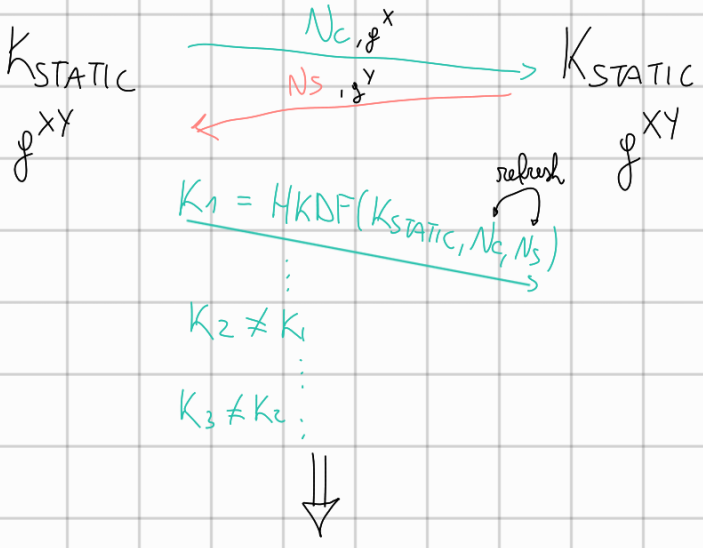
• Come autentificarsi se non ho alcun certificato? metto pre shared key tra client e server, conoscerla è l'autenticazione. (NON PARLO DI CERTIFICATI). NON VOGLIO USARLO PER ENCRYPTION (costante).

1. ▸ è usabile ? (per refresh encryption ?)
2. ▸ è compatibile con PFS ?

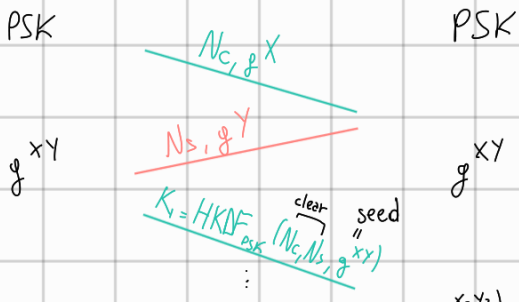
1.) Posso usare pre shared secret da TLS 1.2, per gestione certificati. (es: IoT)

Posso anche salvare info come preshared secret e usarlo per le prossime autenticazioni (evito rinegoziazioni).

TLS 1.3 garantisce ANCHE Forward Secrecy + Pre Shared. Come !?



Non ho forward secrecy (è come DH fixed).
Come posso ottenerla insieme a preshared?
Anonymus DH coefficient. Se uso preshared è perché non posso gestire certificati.
Uso K_{STATIC} (trusted) uso per la derivazione



$K_2 = HKDF_{PSK}(N_{C2}, N_{S2}, g^{X_2Y_2})$

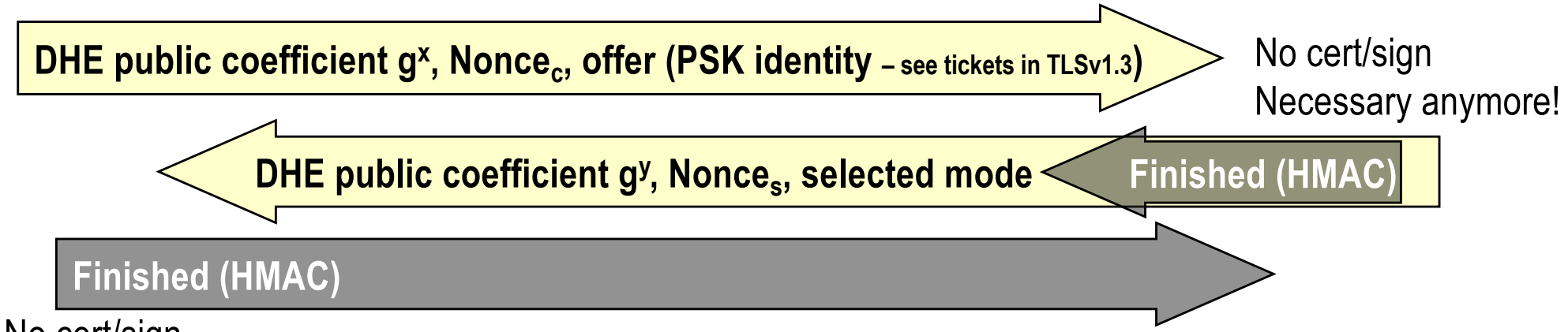
$K_3 = HKDF_{PSK}(N_{C3}, N_{S3}, g^{X_3Y_3})$ ← Breach, trovo PSK, ma serve "x_3" o "y_3", protetto DH exchange, OK passato!

Per FUTURO, non posso monitorare i dati e basta, devo fare ATTACCO ATTIVO per avere $g^{X_3Y_3}$. Nelle app di messaggistica è così (NO CERTIFICATI)

TLS 1.3 handshake:
Further options

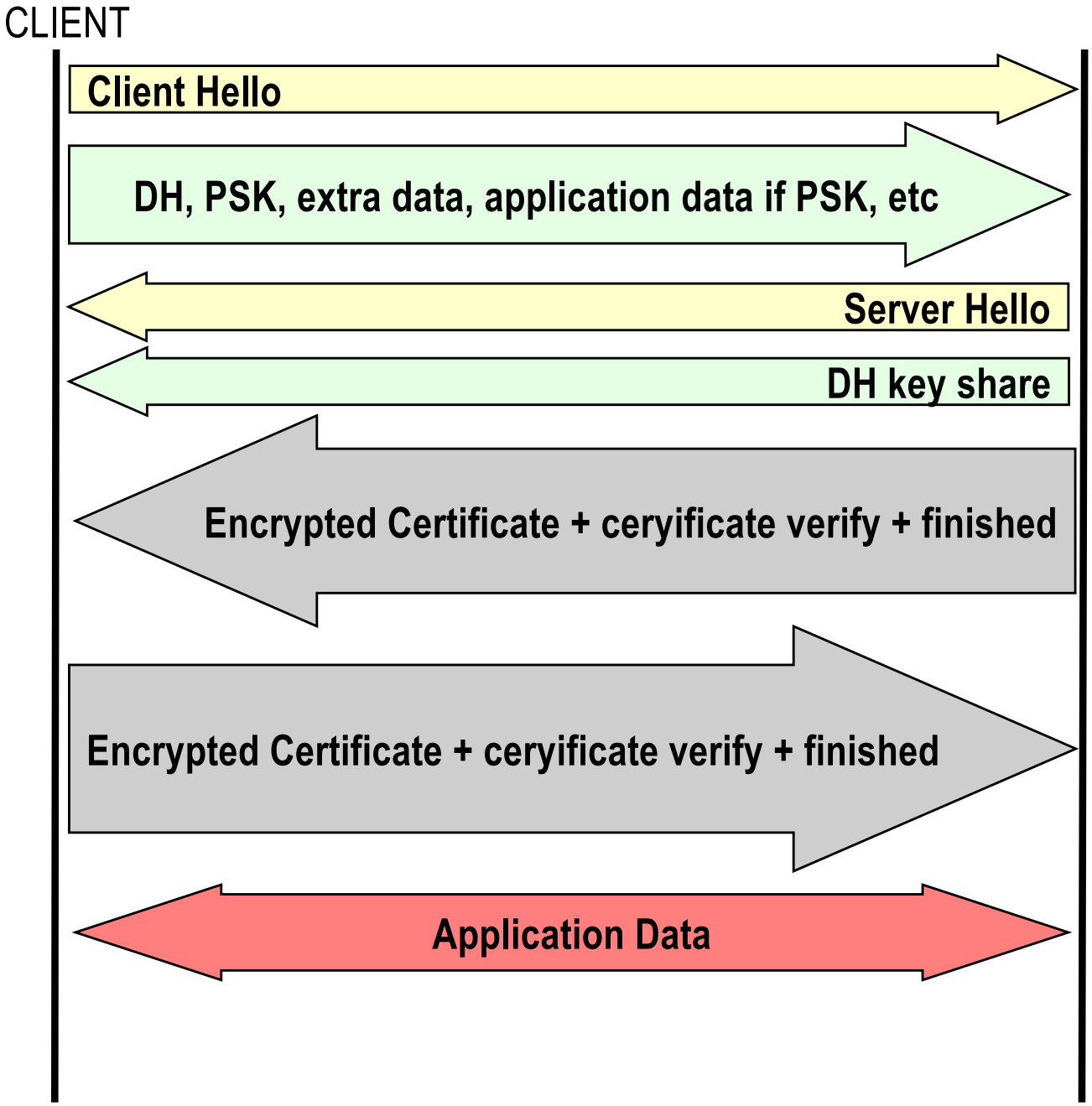
➔Pre-Shared Key

- ⇨PSK, already added in TLS 1.2; PSK is either
  - ➔Agreed offline in constrained environments (IoT)
  - ➔Computed during the first handshake and then reused in subsequent TSL connections
- ⇨New in TLS1.3: optional support for forward secrecy
  - ➔Combines pre-shared key with a fresh DHE



Session keys = HKDF(PSK, g^{xy}, etc) → inclusion of g^{xy} guarantees PFS

New handshake structure – wrap-up



VERY MUCH DIFFERENT!

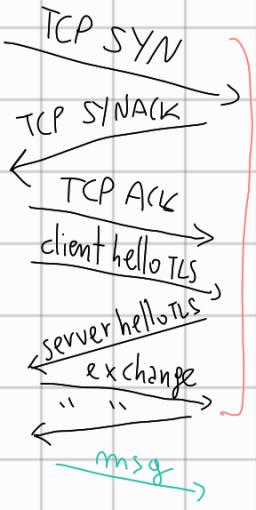
3-way handshake

No more change Cipher specs

Handshake encrypted ASAP

Certificate + cert verify both sides

Ø-RTT



spendo molti RTT
1x handshake TCP
2x handshake TLS

Se avessi pre-shared?

me → Google

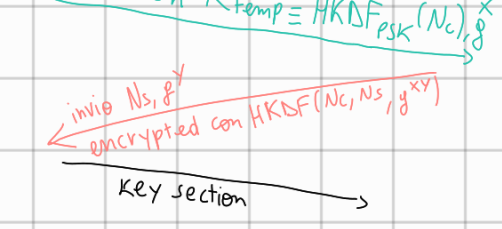
memorizzo temporany PSK da usare dopo per risparmiare RTT

} setup completo per avere PSK

} uso PSK per auth. (partial handshake)

- non serve exchange se PSK è già sull'altro lato. Tutto ciò va contro quello che abbiamo detto, ma realizza Ø-RTT (la pw del router la metto una volta!)

COMPROMESSO



Posso trasferire dati immediamente se accetto che, per un certo tempo (1° round) sono "più vulnerabile" ≡ REPLY ATTACK

RICAPITOLO

- Ø-RTT solo con PSK
- voglio cambiare chiave ∀ sessione

→ derivo $K_{sess} = HKDF(N_c, N_s)$, richiede risposta (cioè N_s)

TLS 1.3 usa PARTIAL NONCE nel 1° msg temporaneo

$K_{Temp} = HKDF(N_c)$, dopo uso quella COMPLETA.
↳ soffre REPLY ATTACK

TLS 1.3 handshake:
Further options

➔**0-RTT Data** (da usare con "cautela")

- ⇨When PSK agreed (in previous handshake), send application data directly in first message
  - ➔Use, as key: $HKDF_{PSK}$(Client-Hello-content)
  - ➔i.e. «mix» PSK with new nonce, offer, and client's DHE public coefficient
- ⇨MUCH faster!
  - ➔Compelling use case: send HTTP request directly in very first message: saves a full RTT!
  - ➔Also, very useful in applications relying on short data transfer
- ⇨**WAY Less security (obvious replay attack): to be used with care… at your own risk!**

Mitigate replay in 0-RTT

- **Problem: key used in first msg does not include server nonce**
- **Time window mitigation**
  - Include maximum lifetime after first session
    - Standard default = 7 days
  - Does not solve… but at least reduces vulnerability period
- **Control nonce reuse**, come riconosco reply attack!?
  - Keep a record of all client nonces
  - Hard to deploy in large scale systems
    - Load balancing, multiple servers, etc…

Other TLS 1.3 nits

take home: less is more in security

➔ **No renegotiation anymore**

⇨ Key Upgrade feature, when rekeying ***inside*** same TLS session is required

➔ **HKDF officially included**

➔ **A few simplification in EC management**

⇨ Single EC point format, no negotiation

➔ **Removal of compression**

⇨ What else? Remember CRIME ☺

➔ **Exported key [integrates RFC 5705]**

⇨ Permits to export to applications a further shared secret key

➔Exporter master secret

➔Prevents application developers to "invent" their own approach!!

⇨ different and independent from the session master key!