

# Apports de TLS 1.3 à la sécurité des communications sur internet

Olivier Levillain

ANSSI

Séminaire SAMOVAR

# Qui suis-je ?

Olivier Levillain

- ▶ @pictyeye sur Twitter
- ▶ <https://paperstreet.picty.org/yeye>
  
- ▶ stage de DEA en cryptographie sur une fonction de hachage
- ▶ membre du laboratoire « système » de l'ANSSI (2007-2012)
- ▶ responsable du laboratoire « réseau » de l'ANSSI (2012-2015)
- ▶ responsable du CFSSI, centre de formation de l'ANSSI (2015-)
  
- ▶ docteur en informatique (2011-2016) : thèse sur SSL/TLS

# Présentation rapide de l'ANSSI

- ▶ L'agence nationale de la sécurité des systèmes d'information, créée en 2009 pour remplacer la DCSSI
- ▶ Son objectif est la prévention et la réaction dans le domaine de la sécurité du numérique
- ▶ L'ANSSI est rattachée au SGDSN, qui dépend du Premier Ministre
- ▶ Effectifs : environ 500 personnes aujourd'hui

## Une brève histoire de SSL/TLS

Plus de deux décennies de vulnérabilités SSL/TLS

Limites de TLS 1.3 et perspectives

Conclusion

# SSL/TLS : un pilier de la sécurité d'Internet

- ▶ Le schéma `https://` inventé par Netscape en 1995
  - ▶ début du commerce en ligne
- ▶ Omniprésence de SSL/TLS aujourd'hui
  - ▶ HTTPS, bien au-delà du commerce en ligne
  - ▶ Une méthode générique pour sécuriser d'autres protocoles (SMTP, IMAP, LDAP...)
  - ▶ VPN SSL
  - ▶ EAP TLS
- ▶ Objectifs de sécurité
  - ▶ Authentification du serveur (et du client)
  - ▶ Protection en confidentialité et en intégrité des données
  - ▶ Anti-rejeu

# Fonctionnement du protocole

Client

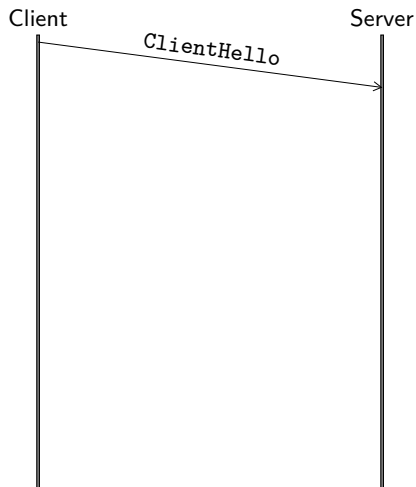


Server



Hypothèses

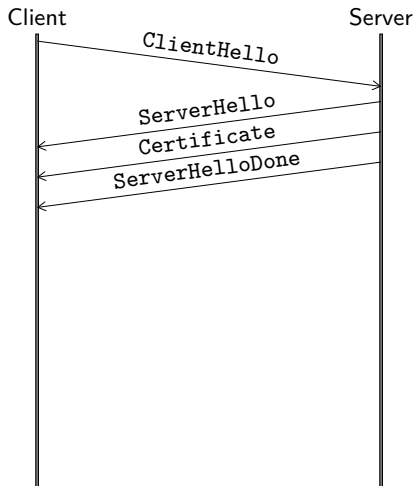
# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1

# Fonctionnement du protocole

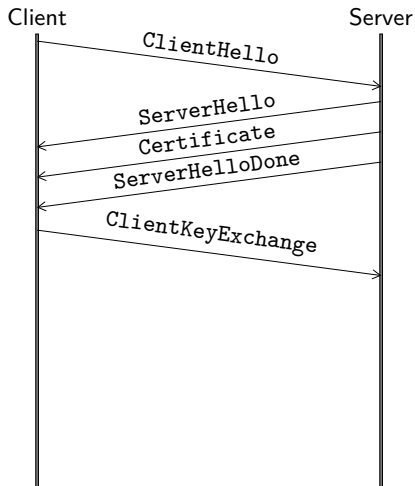


## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA



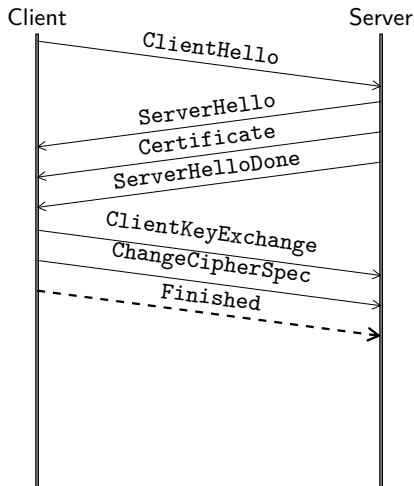
# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA

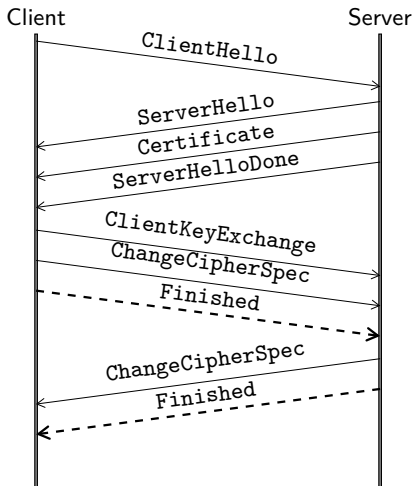
# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA

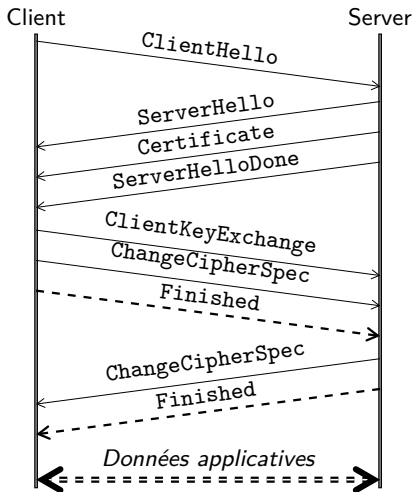
# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA

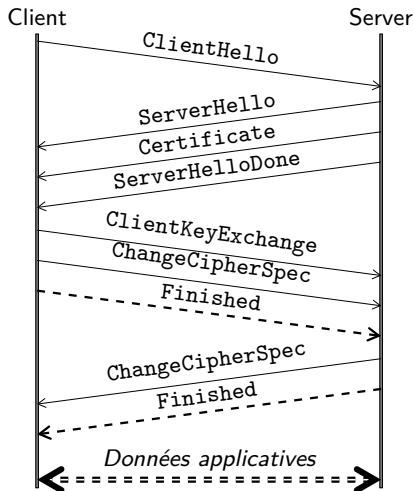
# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA
- ▶ Ni erreur ni incompatibilité

# Fonctionnement du protocole



## Hypothèses

- ▶ Version : SSLv3 - TLS 1.1
- ▶ Algorithme d'échange de clé : par chiffrement RSA
- ▶ Ni erreur ni incompatibilité

## Il existe des variantes

- ▶ échange de clé DHE
- ▶ reprise de session
- ▶ échange de clé symétrique
- ▶ ...

## Versions du protocole



## Versions du protocole

SSLv2

1994

A horizontal black arrow pointing to the right, representing a timeline. A vertical line segment intersects the arrow at the left end. The text 'SSLv2' is positioned above the arrow at this intersection, and the year '1994' is positioned below it.

Création par Netscape de SSL et HTTPS

## Versions du protocole



Face aux nombreuses failles et limitations de SSLv2

- ▶ Netscape SSLv3
- ▶ Microsoft PCT 1.0 (Private Communication Technology)



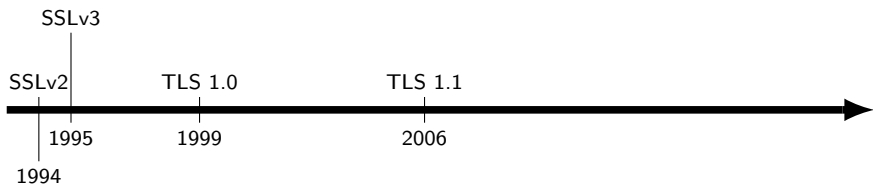
## Versions du protocole



### Standardisation de SSLv3 par l'IETF

- ▶ Démarche entamée en 1996
- ▶ TLS 1.0 = SSLv3.1

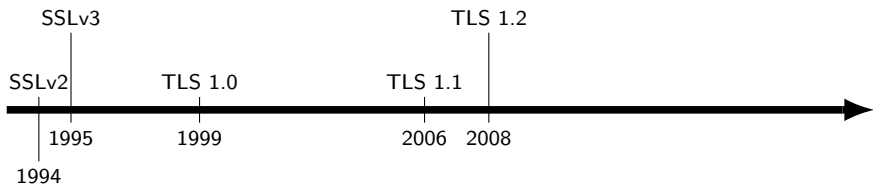
## Versions du protocole



### Correction d'un défaut cryptographique

- ▶ Attaque possible sur le mode CBC avec IV implicite
- ▶ Il s'agit de BEAST, attaque publiée en 2011

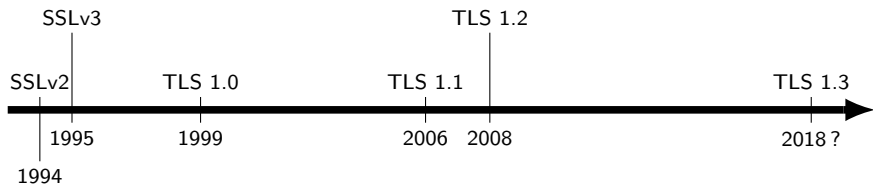
## Versions du protocole



### Modernisation des constructions cryptographiques

- ▶ Apparition des modes combinés (AEAD) avec GCM
- ▶ Apparition de SHA-2
- ▶ Mais les constructions obsolètes sont toujours disponibles

## Versions du protocole



TLS 1.3 : une révolution devenue nécessaire ?

# Usages de SSL/TLS

## Deux modes d'utilisation

- ▶ implicite : le service sécurisé est en écoute sur un autre port que le service en clair
  - ▶ HTTPS (443 vs 80), IMAPS (993 vs 143)...
  - + fonctionnement simple (le dialogue applicatif vient après TLS)
  - besoin d'allouer un port (quid des *firewalls*?)
  - à une époque, incompatibilité avec le *virtual hosting*

# Usages de SSL/TLS

## Deux modes d'utilisation

- ▶ implicite : le service sécurisé est en écoute sur un autre port que le service en clair
  - ▶ HTTPS (443 vs 80), IMAPS (993 vs 143)...
  - + fonctionnement simple (le dialogue applicatif vient après TLS)
  - besoin d'allouer un port (quid des *firewalls*?)
  - à une époque, incompatibilité avec le *virtual hosting*
  
- ▶ explicite : le passage à SSL/TLS se fait pendant le dialogue applicatif
  - ▶ STARTTLS pour SMTP, IMAP et POP
  - ▶ méthode HTTP UPDATE
  - + réutilisation du port
  - complexité de la machine à état
  - mode dégradé en clair plus facile à forcer

## Interlude : quelques chiffres sur SSL/TLS

- ▶ Plus de 50 RFC
- ▶ 5 versions du protocole pour le moment
- ▶ Plus de 300 suites cryptographiques
- ▶ Plus de 20 extensions
- ▶ Quelques fonctionnalités *intéressantes*
  - ▶ compression
  - ▶ renégociation
  - ▶ reprise de session (2 méthodes)
- ▶ Une douzaine d'implémentations bien connues
- ▶ Et combien de piles maison ?

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?



## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

A **AES128-SHA**

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

A **AES128-SHA**

B **ECDH-ECDSA-AES128-SHA**

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA**
- B **ECDH-ECDSA-AES128-SHA**
- C une alerte

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA**
- B **ECDH-ECDSA-AES128-SHA**
- C une alerte
- D la réponse D (**RC4\_MD5**)

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA**
- B **ECDH-ECDSA-AES128-SHA**
- C une alerte
- D la réponse D (**RC4\_MD5**)

Le pire, c'est qu'on peut l'expliquer

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA** (0x002f)
- B **ECDH-ECDSA-AES128-SHA**
- C une alerte
- D la réponse D (**RC4\_MD5**)

Le pire, c'est qu'on peut l'expliquer

- ▶ une suite cryptographique est représentée par un entier sur 16 bits
- ▶ pendant longtemps, toutes les suites étaient de la forme 00 XX

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA** (0x002f)
- B **ECDH-ECDSA-AES128-SHA** (0xc005)
- C une alerte
- D la réponse D (**RC4\_MD5**) (0x0005)

Le pire, c'est qu'on peut l'expliquer

- ▶ une suite cryptographique est représentée par un entier sur 16 bits
- ▶ pendant longtemps, toutes les suites étaient de la forme 00 XX

## Interlude : les piles SSL/TLS maison (1/3)

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : **AES128-SHA** et **ECDH-ECDSA-AES128-SHA** ?

- A **AES128-SHA** (0x002f)
- B **ECDH-ECDSA-AES128-SHA** (0xc005)
- C une alerte
- D la réponse D (**RC4\_MD5**) (0x0005)

Le pire, c'est qu'on peut l'expliquer

- ▶ une suite cryptographique est représentée par un entier sur 16 bits
- ▶ pendant longtemps, toutes les suites étaient de la forme 00 XX
- ▶ pourquoi perdre du temps à regarder l'octet de poids fort ?



## Interlude : les piles SSL/TLS maison (2/3)

- ▶ En 2010, Google propose des extensions, *False Start* et *Snap Start*
- ▶ Après plusieurs mois, Internet se montre intolérant à *Snap Start*
- ▶ L'expérimentation est abandonnée en 2012

## Interlude : les piles SSL/TLS maison (2/3)

- ▶ En 2010, Google propose des extensions, *False Start* et *Snap Start*
- ▶ Après plusieurs mois, Internet se montre intolérant à *Snap Start*
- ▶ L'expérimentation est abandonnée en 2012
  
- ▶ Un an plus tard, le même problème resurgit dans un autre contexte
- ▶ Sur la liste de diffusion IETF ([tls@ietf.org](mailto:tls@ietf.org)), on apprend finalement la raison : le `ClientHello` est trop gros...

## Interlude : les piles SSL/TLS maison (3/3)

Examinons le début d'un ClientHello de 258 octets

16	03	01	01	02
----	----	----	----	----

TLS	Type	Version	Longueur
	<i>HS</i>	<i>TLS 1.0</i>	<i>258</i>

SSLv2	Longueur	<i>Pad.</i>	Type
	<i>5635</i>	<i>...</i>	<i>CH</i>

## Interlude : les piles SSL/TLS maison (3/3)

Examinons le début d'un `ClientHello` de 258 octets

16	03	01	01	02
----	----	----	----	----

TLS	Type	Version	Longueur
	<i>HS</i>	<i>TLS 1.0</i>	<i>258</i>

SSLv2	Longueur	<i>Pad.</i>	Type
	<i>5635</i>	<i>...</i>	<i>CH</i>

Un `ClientHello` TLS dont la taille est comprise entre 256 et 511 peut être confondu avec un `ClientHello` SSLv2 !

## Interlude : les piles SSL/TLS maison (3/3)

Examinons le début d'un `ClientHello` de 258 octets

16	03	01	01	02
----	----	----	----	----

TLS	Type	Version	Longueur
	<i>HS</i>	<i>TLS 1.0</i>	<i>258</i>

SSLv2	Longueur	<i>Pad.</i>	Type
	<i>5635</i>	<i>...</i>	<i>CH</i>

Un `ClientHello` TLS dont la taille est comprise entre 256 et 511 peut être confondu avec un `ClientHello` SSLv2 !

Tout est bien qui finit bien

- ▶ Google a finalement proposé une extension pour ajouter du bourrage au `ClientHello`...

Une brève histoire de SSL/TLS

Plus de deux décennies de vulnérabilités SSL/TLS

Limites de TLS 1.3 et perspectives

Conclusion

## Classification des vulnérabilités

De nombreuses failles et attaques sur SSL/TLS depuis 1995

Une proposition de classification

- ▶ failles du *Handshake Protocol*
- ▶ attaques contre le *Record Protocol*
- ▶ problèmes liés aux certificats
- ▶ erreurs d'implémentations

Limites de l'exercice

- ▶ certaines failles relèvent de plusieurs catégories
- ▶ certains étiquetages sont discutables

## Classification des vulnérabilités

De nombreuses failles et attaques sur SSL/TLS depuis 1995

Une proposition de classification

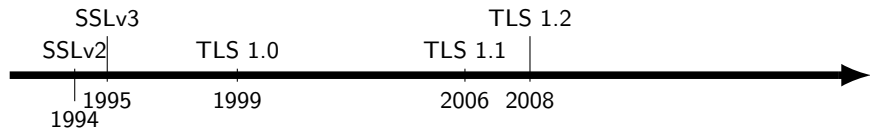
- ▶ failles du *Handshake Protocol*
- ▶ attaques contre le *Record Protocol*
- ▶ problèmes liés aux certificats
- ▶ erreurs d'implémentations

Limites de l'exercice

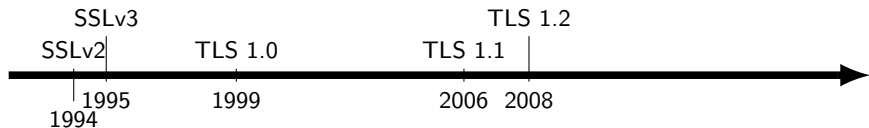
- ▶ certaines failles relèvent de plusieurs catégories
- ▶ certains étiquetages sont discutables



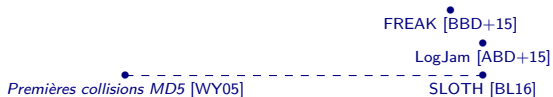
## Failles du *Handshake Protocol*



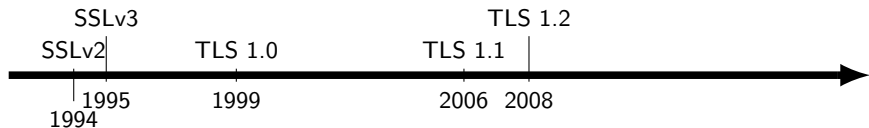
## Failles du *Handshake Protocol*



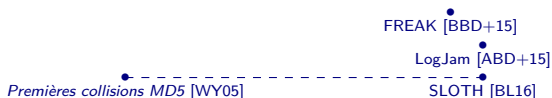
.....  
*Paramètres crypto faibles*



## Failles du *Handshake Protocol*



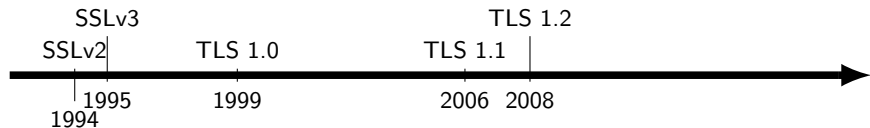
*Paramètres crypto faibles*



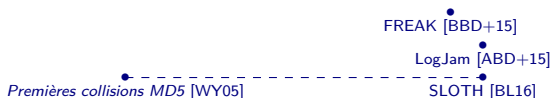
*Failles de spécification*



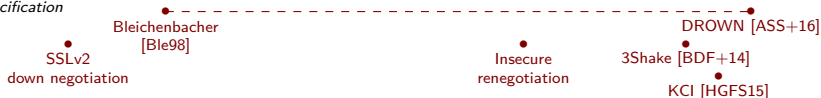
## Failles du *Handshake Protocol*



Paramètres crypto faibles



Failles de spécification



Cross-protocol attacks



## Paramètres cryptographiques faibles

Jusqu'à peu, des piles TLS récentes acceptaient de négocier

- ▶ MD5 (SLOTH)
- ▶ RSA-EXPORT (FREAK)
- ▶ des groupes Diffie-Hellman de 512 bits (LogJam)

## Paramètres cryptographiques faibles

Jusqu'à peu, des piles TLS récentes acceptaient de négocier

- ▶ MD5 (SLOTH)
- ▶ RSA-EXPORT (FREAK)
- ▶ des groupes Diffie-Hellman de 512 bits (LogJam)

Avec TLS 1.3, tout cela disparaît

- ▶ seules des fonctions de hachage récentes subsistent\*
- ▶ les échanges de clé RSA et RSA-EXPORT disparaissent au profit d'échanges Diffie-Hellman\*\* ...
- ▶ ... sur des groupes nommés de taille raisonnable

## Paramètres cryptographiques faibles

Jusqu'à peu, des piles TLS récentes acceptaient de négocier

- ▶ MD5 (SLOTH)
- ▶ RSA-EXPORT (FREAK)
- ▶ des groupes Diffie-Hellman de 512 bits (LogJam)

Avec TLS 1.3, tout cela disparaît

- ▶ seules des fonctions de hachage récentes subsistent\*
- ▶ les échanges de clé RSA et RSA-EXPORT disparaissent au profit d'échanges Diffie-Hellman\*\* ...
- ▶ ... sur des groupes nommés de taille raisonnable

Limites

- \* TLS 1.3 accepte encore SHA-1 dans les certificats
- \*\* Les clés RSA de signature peuvent toujours être minuscules

## Erreurs de conception (1/2)

### Quelques exemples d'échecs

- ▶ Négociation à la baisse dans SSLv2 : seules les valeurs aléatoires échangés étaient authentifiées
- ▶ Vulnérabilité sur la renégociation / *Triple Handshake* : les différentes sessions/*époques* n'étaient pas liées de manière suffisante



## Erreurs de conception (1/2)

### Quelques exemples d'échecs

- ▶ Négociation à la baisse dans SSLv2 : seules les valeurs aléatoires échangés étaient authentifiées
- ▶ Vulnérabilité sur la renégociation / *Triple Handshake* : les différentes sessions/*époques* n'étaient pas liées de manière suffisante

Jusqu'à TLS 1.2, le *master secret* est dérivé à partir

- ▶ du résultat de l'algorithme d'échange de clé
- ▶ des valeurs aléatoires échangées en clair

L'intégrité n'est garantie qu'a posteriori, avec les messages `Finished`

## Erreurs de conception (1/2)

### Quelques exemples d'échecs

- ▶ Négociation à la baisse dans SSLv2 : seules les valeurs aléatoires échangés étaient authentifiées
- ▶ Vulnérabilité sur la renégociation / *Triple Handshake* : les différentes sessions/*époques* n'étaient pas liées de manière suffisante

Jusqu'à TLS 1.2, le *master secret* est dérivé à partir

- ▶ du résultat de l'algorithme d'échange de clé
- ▶ des valeurs aléatoires échangées en clair

L'intégrité n'est garantie qu'a posteriori, avec les messages `Finished`

Ce n'est *pas* suffisant (3SHAKE, FREAK, etc.).

## Erreurs de conception (2/2)

Avec TLS 1.3, tout le transcript est signé

- ▶ erreur crypto classique : il faut tout signer de manière explicite
- ▶ de nombreuses attaques deviennent caduques\*

## Erreurs de conception (2/2)

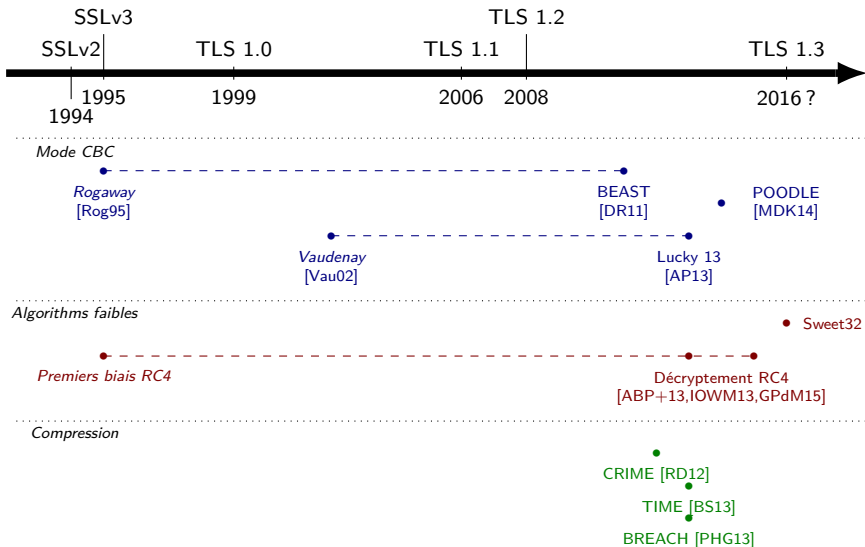
Avec TLS 1.3, tout le transcript est signé

- ▶ erreur crypto classique : il faut tout signer de manière explicite
- ▶ de nombreuses attaques deviennent caduques\*

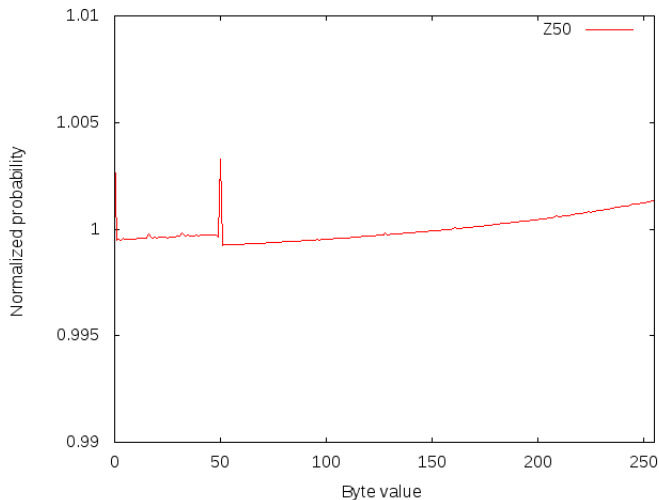
Limites

- \* Il faut encore prendre en compte les attaques forçant une version antérieure de TLS

# Attaques contre *Record Protocol*



# Zoom sur RC4



# Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller

## Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller

- ▶ Le mode CBC dans TLS favorise les oracles de padding
- ▶ L'utilisation d'un IV implicite peut mener à des attaques
- ▶ Avec SSLv3, les octets de padding sont mal spécifiés, ce qui amplifie les oracles



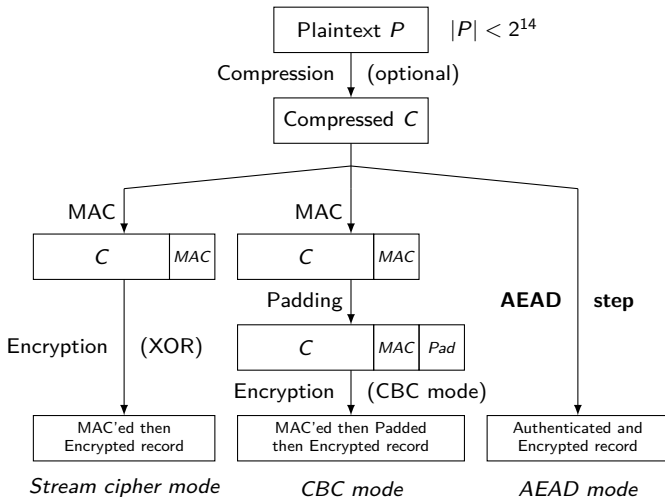
## Réflexions autour du mode CBC

`https://www.openssl.org/~bodo/tls-cbc.txt`

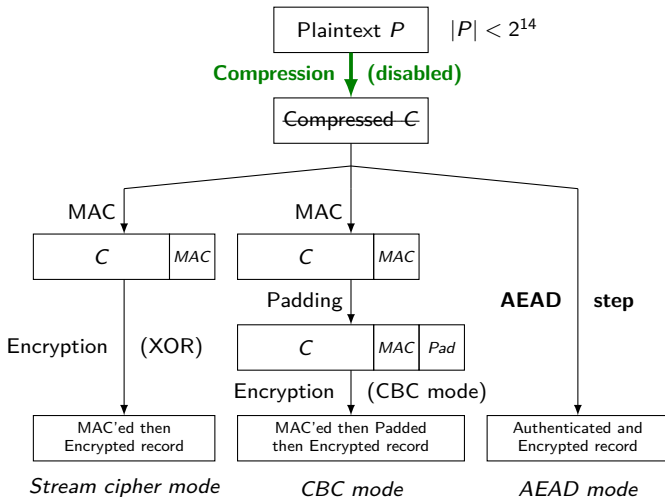
Bodo Möller (2004-05-20)

- ▶ Le mode CBC dans TLS favorise les oracles de padding
- ▶ L'utilisation d'un IV implicite peut mener à des attaques
- ▶ Avec SSLv3, les octets de padding sont mal spécifiés, ce qui amplifie les oracles
- ▶ ... il décrivait Lucky 13, BEAST et POODLE

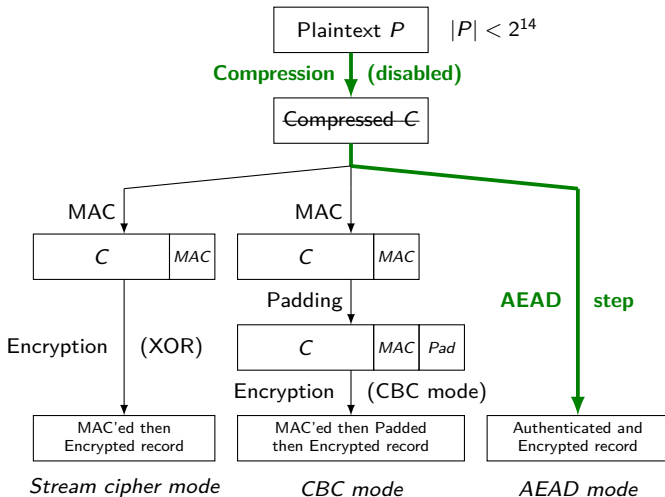
## Record Protocol : la solution



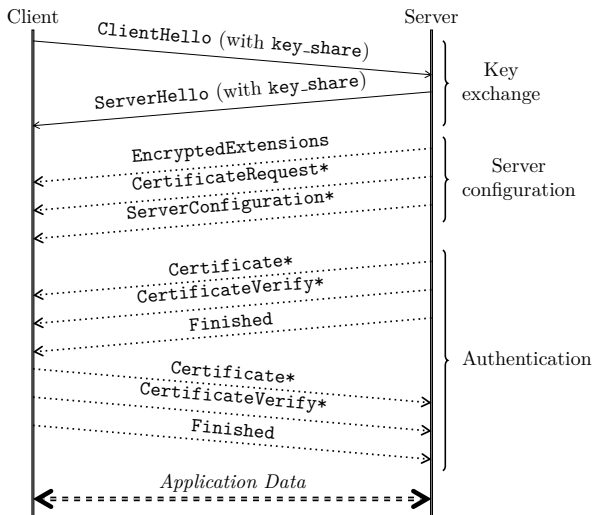
# Record Protocol : la solution



# Record Protocol : la solution



# TLS 1.3 : flot de messages



## TLS 1.3 : la solution à tous les problèmes ?

Hypothèses pour garantir les objectifs de sécurité

- ▶ utiliser des algorithmes et des clés décentes dans les certificats
- ▶ refuser des paramètres faibles
- ▶ question plus générale de la confiance dans l'IGC

## TLS 1.3 : la solution à tous les problèmes ?

### Hypothèses pour garantir les objectifs de sécurité

- ▶ utiliser des algorithmes et des clés décentes dans les certificats
- ▶ refuser des paramètres faibles
- ▶ question plus générale de la confiance dans l'IGC

### Hypothèses si on doit vivre avec des versions antérieures

- ▶ TLS 1.2 minimum
- ▶ échange de clé ECDHE (à cause des mécanismes d'*anti-downgrade* et de LogJam)
- ▶ mode AEAD

## TLS 1.3 : la solution à tous les problèmes ?

Hypothèses pour garantir les objectifs de sécurité

- ▶ utiliser des algorithmes et des clés décentes dans les certificats
- ▶ refuser des paramètres faibles
- ▶ question plus générale de la confiance dans l'IGC

Hypothèses si on doit vivre avec des versions antérieures

- ▶ TLS 1.2 minimum
- ▶ échange de clé ECDHE (à cause des mécanismes d'*anti-downgrade* et de LogJam)
- ▶ mode AEAD

Quid de l'implémentation ?



Une brève histoire de SSL/TLS

Plus de deux décennies de vulnérabilités SSL/TLS

Limites de TLS 1.3 et perspectives

Conclusion

## Failles d'implémentation sur TLS

2014 : une année dure pour les piles TLS

- ▶ février : `goto fail` (Apple)
- ▶ février : `goto fail` (GnuTLS)
- ▶ avril : *Heartbleed* (OpenSSL)
- ▶ juin : *Early CCS* (OpenSSL)
- ▶ septembre : Universal signature forgery (Berserk ?) (Mozilla NSS)
- ▶ novembre : exécution de code arbitraire (MS SChannel)

# Faibles d'implémentation sur TLS

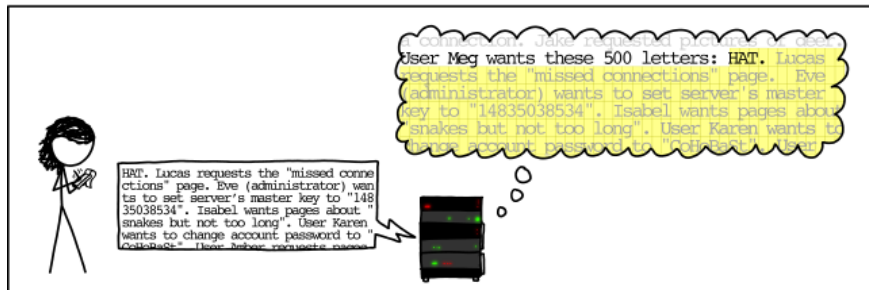
2014 : une année dure pour les piles TLS

- ▶ février : `goto fail` (Apple)
- ▶ février : `goto fail` (GnuTLS)
- ▶ avril : *Heartbleed* (OpenSSL)
- ▶ juin : *Early CCS* (OpenSSL)
- ▶ septembre : Universal signature forgery (Berserk ?) (Mozilla NSS)
- ▶ novembre : exécution de code arbitraire (MS SChannel)

Catégories de failles

- ▶ les erreurs classiques
- ▶ les problèmes de *parsers*
- ▶ le coût de la crypto obsolète
- ▶ la complexité des machines à état

## Erreurs classiques (1/3) : Heartbleed



Source : <http://xkcd.com/1354>

```
+      /* Read type and payload length first */
+      if (1 + 2 + 16 > s->s3->rrec.length)
+          return 0; /* silently discard */
```

## Erreurs classiques (2/3) : WinShock

- ▶ L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- ▶ Le message `Certificate`, qui contient la chaîne de certification
  - ▶ il contient la courbe utilisée
  - ▶ en particulier, il indique la taille  $L$  du corps sous-jacent
- ▶ `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - ▶ la signature contient les coordonnées d'un point, d'une taille  $l$
  - ▶ supposons que `SChannel` copie  $l$  octets dans une zone allouée pour  $L$  octets, sans se poser de question...

## Erreurs classiques (2/3) : WinShock

- ▶ L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- ▶ Le message `Certificate`, qui contient la chaîne de certification
  - ▶ il contient la courbe utilisée
  - ▶ en particulier, il indique la taille  $L$  du corps sous-jacent
- ▶ `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - ▶ la signature contient les coordonnées d'un point, d'une taille  $l$
  - ▶ supposons que `SChannel` copie  $l$  octets dans une zone allouée pour  $L$  octets, sans se poser de question...

Sauf que l'authentification client par certificat est rarement utilisée

## Erreurs classiques (2/3) : WinShock

- ▶ L'authentification client utilisant des certificats avec des courbes elliptiques repose sur deux messages
- ▶ Le message `Certificate`, qui contient la chaîne de certification
  - ▶ il contient la courbe utilisée
  - ▶ en particulier, il indique la taille  $L$  du corps sous-jacent
- ▶ `CertificateVerify`, qui contient une signature couvrant les messages précédents
  - ▶ la signature contient les coordonnées d'un point, d'une taille  $I$
  - ▶ supposons que `SChannel` copie  $I$  octets dans une zone allouée pour  $L$  octets, sans se poser de question...

Sauf que l'authentification client par certificat est rarement utilisée

*Teaser* : tous les serveurs vulnérables étaient pourtant exploitables

## Erreurs classiques (3/3) : True, False, FILE\_NOT\_FOUND

Analysons la faille CVE-2014-0092 de GnuTLS (mars 2014) :

*But this bug is arguably much worse than APPLE's, as it has allowed crafted certificates to evade validation check for all versions of GNUTLS ever released since that project got started in late 2000.[...]*

*The `check_if_ca` function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that **error returns were misinterpreted as certificate validations**.*



## Erreurs classiques (3/3) : True, False, FILE\_NOT\_FOUND

Analysons la faille CVE-2014-0092 de GnuTLS (mars 2014) :

*But this bug is arguably much worse than APPLE's, as it has allowed crafted certificates to evade validation check for all versions of GNUTLS ever released since that project got started in late 2000.[...]*

*The `check_if_ca` function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that **error returns were misinterpreted as certificate validations**.*

Au passage, une faille similaire avait été trouvée dans OpenSSL... en 2008 (CVE-2008-5077).

## Erreurs classiques (3/3) : True, False, FILE\_NOT\_FOUND

Analysons la faille CVE-2014-0092 de GnuTLS (mars 2014) :

*But this bug is arguably much worse than APPLE's, as it has allowed crafted certificates to evade validation check for all versions of GNUTLS ever released since that project got started in late 2000.[...]*

*The `check_if_ca` function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that **error returns were misinterpreted as certificate validations**.*

Au passage, une faille similaire avait été trouvée dans OpenSSL... en 2008 (CVE-2008-5077).

Il est peut-être temps de considérer que les développeurs ne sont pas seuls responsables... et d'améliorer nos langages/outils/spécifications

## Le coût de la crypto obsolète (1/3) : Bleichenbacher

### RSA PKCS#1 v1.5

- ▶ le chiffrement RSA repose sur un schéma de bourrage
- ▶ comment traiter un *padding* invalide lors du déchiffrement ?

## Le coût de la crypto obsolète (1/3) : Bleichenbacher

### RSA PKCS#1 v1.5

- ▶ le chiffrement RSA repose sur un schéma de bourrage
- ▶ comment traiter un *padding* invalide lors du déchiffrement ?

### L'attaque de Bleichenbacher (1998)

- ▶ le principe : envoyer un texte chiffré altéré
- ▶ si l'attaquant peut distinguer un *padding* valide d'un *padding* invalide, il obtient de l'information sur le clair
- ▶ application à TLS : l'attaque *Million Message Attack*

## Le coût de la crypto obsolète (2/3) : Bleichenbacher

L'attaque resurgit en 2014

- ▶ en Java, une erreur de *padding* provoque une exception
- ▶ ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- ▶ un développeur TLS doit choisir entre la modularité et la sécurité

## Le coût de la crypto obsolète (2/3) : Bleichenbacher

L'attaque resurgit en 2014

- ▶ en Java, une erreur de *padding* provoque une exception
- ▶ ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- ▶ un développeur TLS doit choisir entre la modularité et la sécurité

Et en 2016...

- ▶ DROWN (*Decrypting RSA with Obsolete and Weakened eNcryption*)
- ▶ attaque sur SSLv2 pour retrouver des *pre-master secret* TLS
- ▶ des *bugs* dans OpenSSL rendent l'attaque très efficace

## Le coût de la crypto obsolète (2/3) : Bleichenbacher

L'attaque resurgit en 2014

- ▶ en Java, une erreur de *padding* provoque une exception
- ▶ ainsi, pour éviter une *timing attack*, il faut redévelopper l'algorithme
- ▶ un développeur TLS doit choisir entre la modularité et la sécurité

Et en 2016...

- ▶ DROWN (*Decrypting RSA with Obsolete and Weakened eNcryption*)
- ▶ attaque sur SSLv2 pour retrouver des *pre-master secret* TLS
- ▶ des *bugs* dans OpenSSL rendent l'attaque très efficace
- ▶ heureusement que SSLv2 est considéré comme obsolète depuis plus de 10 ans...

## Le coût de la crypto obsolète (3/3) : MAC-then-Encrypt

Les oracles de *padding* existent aussi en symétrique

- ▶ *MAC-then-CBC* est naturellement vulnérable
- ▶ 2002 : Vaudenay présente le principe de l'attaque
- ▶ 2011 : *XML Encryption is broken*
- ▶ 2013 : Lucky 13 (l'attaque est applicable à TLS)



## Le coût de la crypto obsolète (3/3) : MAC-then-Encrypt

Les oracles de *padding* existent aussi en symétrique

- ▶ *MAC-then-CBC* est naturellement vulnérable
- ▶ 2002 : Vaudenay présente le principe de l'attaque
- ▶ 2011 : *XML Encryption is broken*
- ▶ 2013 : Lucky 13 (l'attaque est applicable à TLS)

Le correctif ?

- ▶ du sparadrap (une note d'implémentation dans le standard)
- ▶ un *patch* sordide pour garantir un déchiffrement en temps constant
- ▶ utiliser *Encrypt-then-MAC* (RFC 7366) ou du chiffrement authentifié (AEAD)

Là encore, il faut choisir entre modularité et sécurité (et interopérabilité)

## TLS dans tous ses états (1/2) : SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- ▶ En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)

## TLS dans tous ses états (1/2) : SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- ▶ En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- ▶ Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés

## TLS dans tous ses états (1/2) : SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- ▶ En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- ▶ Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés
- ▶ OpenSSL acceptait l'échange de clé RSA-EXPORT lorsque le chiffrement RSA a été négocié (FREAK)

## TLS dans tous ses états (1/2) : SMACK et FREAK

En 2015, des attaques concernant presque toutes les piles TLS

- ▶ En Java, l'envoi d'un message `Finished` précoce permettait à un attaquant de sauter toute la négociation (dont l'authentification)
- ▶ Dans certaines attaques, il y avait confusion entre les points utilisés par ECDHE et ECDSA lorsque certains messages sont supprimés
- ▶ OpenSSL acceptait l'échange de clé RSA-EXPORT lorsque le chiffrement RSA a été négocié (FREAK)

En général, la machine à états réagit aux messages reçus, au lieu de maintenir une liste restreinte des messages licites

## TLS dans tous ses états (2/2)

### Autres exemples

- ▶ *Early CCS*, trouvé à l'aide de méthodes formelles
- ▶ Dans la vulnérabilité SChannel précédente, les messages `Certificate` et `CertificateVerify` étaient toujours interprétés, même s'ils n'étaient pas sollicités

Presque toutes les piles TLS étaient vulnérables à des attaques similaires :

- ▶ Les specs sont peut-être trop complexes ?
- ▶ Nous avons sans doute besoin de plus de tests

## Sécurité des implémentations TLS 1.3? (1/2)

### Erreurs de programmation classiques

- ▶ la nouvelle spécification n'apporte rien
- ▶ il faut améliorer les méthodes de développement

# Sécurité des implémentations TLS 1.3? (1/2)

## Erreurs de programmation classiques

- ▶ la nouvelle spécification n'apporte rien
- ▶ il faut améliorer les méthodes de développement

## *Parsers*

- ▶ petite amélioration avec TLS 1.3 (moins d'options)...
- ▶ ... mais les messages pourraient ne pas dépendre du contexte
- ▶ ... mais les certificats X.509 sont toujours une source là



## Sécurité des implémentations TLS 1.3? (2/2)

### Cryptographie obsolète

- ▶ en dehors des certificats, TLS 1.3 a fait un gros nettoyage
- ▶ (avec au moins 10 ans de retard)

## Sécurité des implémentations TLS 1.3? (2/2)

### Cryptographie obsolète

- ▶ en dehors des certificats, TLS 1.3 a fait un gros nettoyage
- ▶ (avec au moins 10 ans de retard)

### Machines à état

- ▶ TLS 1.3 *vanilla* beaucoup plus simple
- ▶ ... mais quid de 0 RTT ?
- ▶ ... mais quid de l'authentification tardive du client par certificat ?
- ▶ ... mais quid des messages de compatibilité pour les *middle boxes* ?

Une brève histoire de SSL/TLS

Plus de deux décennies de vulnérabilités SSL/TLS

Limites de TLS 1.3 et perspectives

**Conclusion**

## TLS 1.3 : un nouvel espoir

Parmi les failles présentées, nombreuses sont celles issues de la conception

- ▶ la crypto obsolète, retirée dans TLS 1.3 : PKCS#1 v1.5, RC4, CBC...
- ▶ certains problèmes de l'automate : la négociation a été repensée

Cependant, TLS 1.3 ne résout pas

- ▶ les défauts dans les méthodes de développement
- ▶ les soucis de compatibilité : les versions précédentes de TLS sont encore là pour un certain temps
- ▶ la complexité du standard
  - ▶ X.509
  - ▶ 0-RTT
  - ▶ l'authentification tardive du client
  - ▶ des astuces sordides pour contrer les *middle boxes*

## Langages et méthodologie

Nos langages de programmation devraient nous aider

- ▶ le typage fort permet d'éviter des erreurs simples
- ▶ la gestion de la mémoire peut être sûre

## Langages et méthodologie

Nos langages de programmation devraient nous aider

- ▶ le typage fort permet d'éviter des erreurs simples
- ▶ la gestion de la mémoire peut être sûre

De bons outils sont nécessaires

- ▶ il faut activer les *warnings*
- ▶ et les corriger (`-Werror`)

## Langages et méthodologie

Nos langages de programmation devraient nous aider

- ▶ le typage fort permet d'éviter des erreurs simples
- ▶ la gestion de la mémoire peut être sûre

De bons outils sont nécessaires

- ▶ il faut activer les *warnings*
- ▶ et les corriger (`-Werror`)

Nous avons besoin de tests

- ▶ tests de non-régression
- ▶ tests négatifs

## Langages et méthodologie

Nos langages de programmation devraient nous aider

- ▶ le typage fort permet d'éviter des erreurs simples
- ▶ la gestion de la mémoire peut être sûre

De bons outils sont nécessaires

- ▶ il faut activer les *warnings*
- ▶ et les corriger (`-Werror`)

Nous avons besoin de tests

- ▶ tests de non-régression
- ▶ tests négatifs

Effort de modélisation et de tests en utilisant des méthodes formelles (TRON en 2016, TLS-DIV en 2017)



# Déploiement et perspectives

## Utilisation de TLS

- ▶ standardisation de TLS 1.3 (2018 ?)
- ▶ interdiction progressive (mais rapide) de TLS 1.0 et 1.1
- ▶ restriction des fonctionnalités complexes
- ▶ mesures pour évaluer la maturité de l'écosystème

# Déploiement et perspectives

## Utilisation de TLS

- ▶ standardisation de TLS 1.3 (2018 ?)
- ▶ interdiction progressive (mais rapide) de TLS 1.0 et 1.1
- ▶ restriction des fonctionnalités complexes
- ▶ mesures pour évaluer la maturité de l'écosystème

## Travail sur « tout le reste »

- ▶ la confiance dans les certificats
- ▶ l'interaction entre TLS et les couches applicatives
- ▶ les manières de contourner TLS
- ▶ les couches applicatives...

# Questions ?

Merci pour votre attention

`olivier.levillain@ssi.gouv.fr`

`https://paperstreet.picty.org/yeye`