TLS: passé, présent et futur?

Olivier Levillain

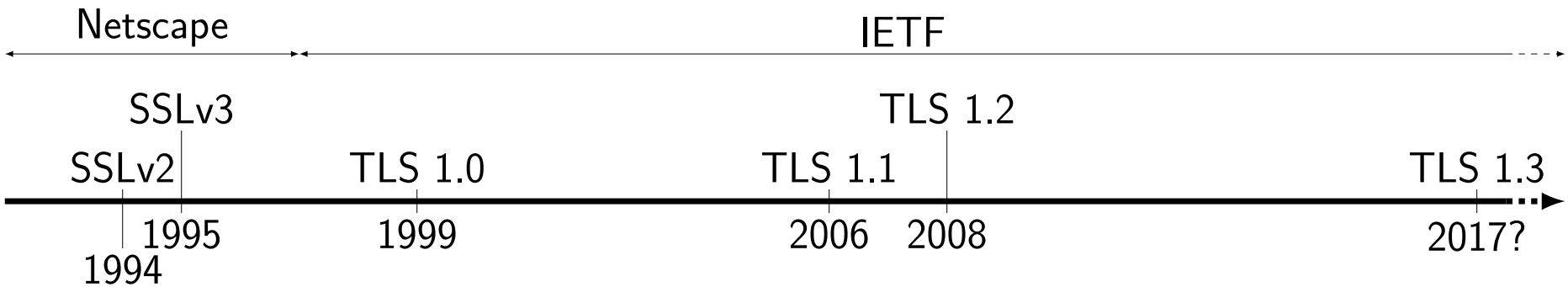
ANSSI

Journées Nationales 2017 Pré-GDR Sécurité Informatique
1er juin 2017

Description rapide de SSL/TLS (1/2)

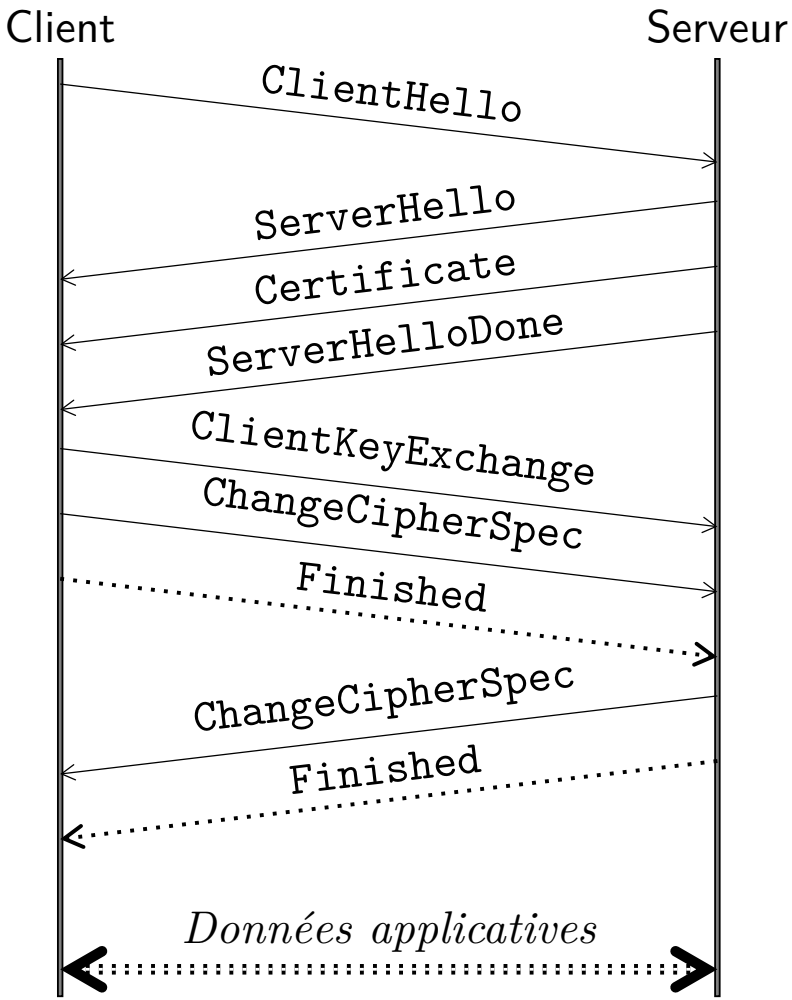
SSL/TLS, un protocole de sécurité garantissant

- l'authentification du serveur (et optionnellement du client)
- la protection en confidentialité et en intégrité des données



Un protocole âgé de plus de 20 ans, utilisé presque partout

Description rapide de SSL/TLS (2/2)



Deux phases

- Handshake Protocol
  - négociation des algorithmes
  - authentification du serveur
  - échange de clé
- Record Protocol
  - échanges de données

Quid de la sécurité du protocole ?

Depuis 1995, beaucoup de failles et d'attaques ont été publiées

On peut les trier selon les catégories suivantes :

- vulnérabilités affectant le *Handshake Protocol*
- attaques contre le *Record Protocol*
- erreurs d'implémentation
- *problèmes liés aux certificats*

Quid de la sécurité du protocole ?

Depuis 1995, beaucoup de failles et d'attaques ont été publiées

On peut les trier selon les catégories suivantes :

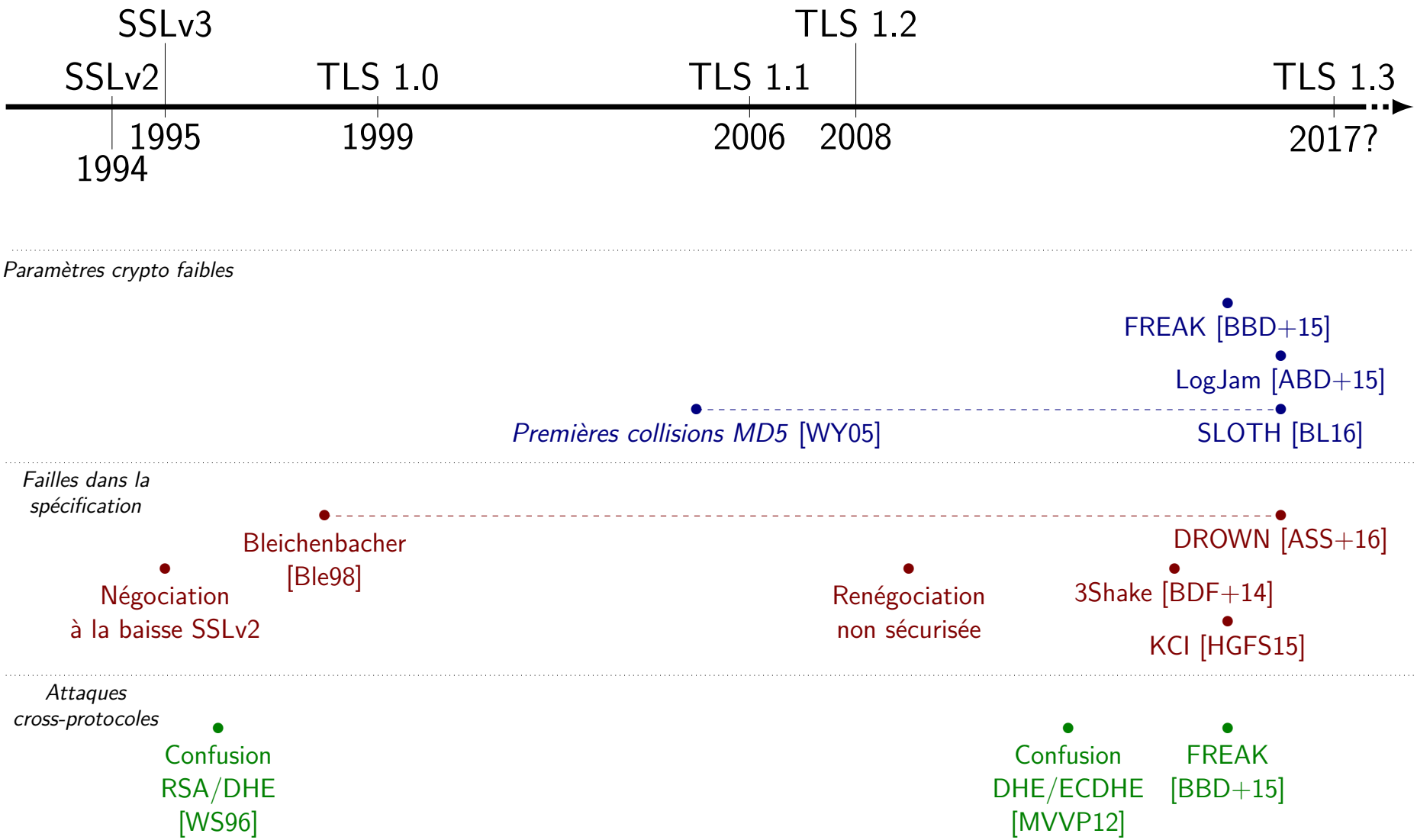
- vulnérabilités affectant le *Handshake Protocol*
- attaques contre le *Record Protocol*
- erreurs d'implémentation
- *problèmes liés aux certificats*

Et demain ?

- enseignement tirés de ces défauts de sécurité ?
- apport de TLS 1.3 face à ces vulnérabilités ?

Vulnérabilités affectant le *Handshake Protocol*

Vulnérabilités affectant le Handshake Protocol



RSA PKCS#1 v1.5

Limitations du chiffrement RSA comme méthode d'échange de clé

- absence de *forward secrecy*
- primitive fragile (oracle décrit par Bleichenbacher en 1998)
- forme d'authentification implicite

RSA PKCS#1 v1.5

Limitations du chiffrement RSA comme méthode d'échange de clé

- absence de *forward secrecy*
- primitive fragile (oracle décrit par Bleichenbacher en 1998)
- forme d'authentification implicite

Solution

- [TLS 1.3] forcer l'utilisation de (EC)DHE pour l'échange de clé

Paramètres et algorithmes faibles

Faiblesses dans SSL/TLS

- fonctions de hachage vulnérables aux collisions (MD5, SHA1)
  - problème évident pour les certificats
  - faille sur le protocole également (SLOTH / collision de transcripts)
- taille de clés faibles
- groupe Diffie-Hellman faibles

Paramètres et algorithmes faibles

Faiblesses dans SSL/TLS

- fonctions de hachage vulnérables aux collisions (MD5, SHA1)
  - problème évident pour les certificats
  - faille sur le protocole également (SLOTH / collision de transcripts)
- taille de clés faibles
- groupe Diffie-Hellman faibles

Solution

- [TLS 1.3] abandonner les fonctions obsolètes
- [TLS 1.3] utiliser uniquement des groupes nommés de taille raisonnable
- ajouter des vérifications sur la taille des clés

Couverture du transcript dans le protocole

Preuve d'intégrité du transcript partielle et tardive

- SSLv2 n'incluait que les aléas dans le message `Finished`
- SSLv3 et TLS répètent la même erreur dans la dérivation de clé et la signature
- cette couverture incomplète rend possibles des attaques par confusion possibles (LogJam)

Couverture du transcript dans le protocole

Preuve d'intégrité du transcript partielle et tardive

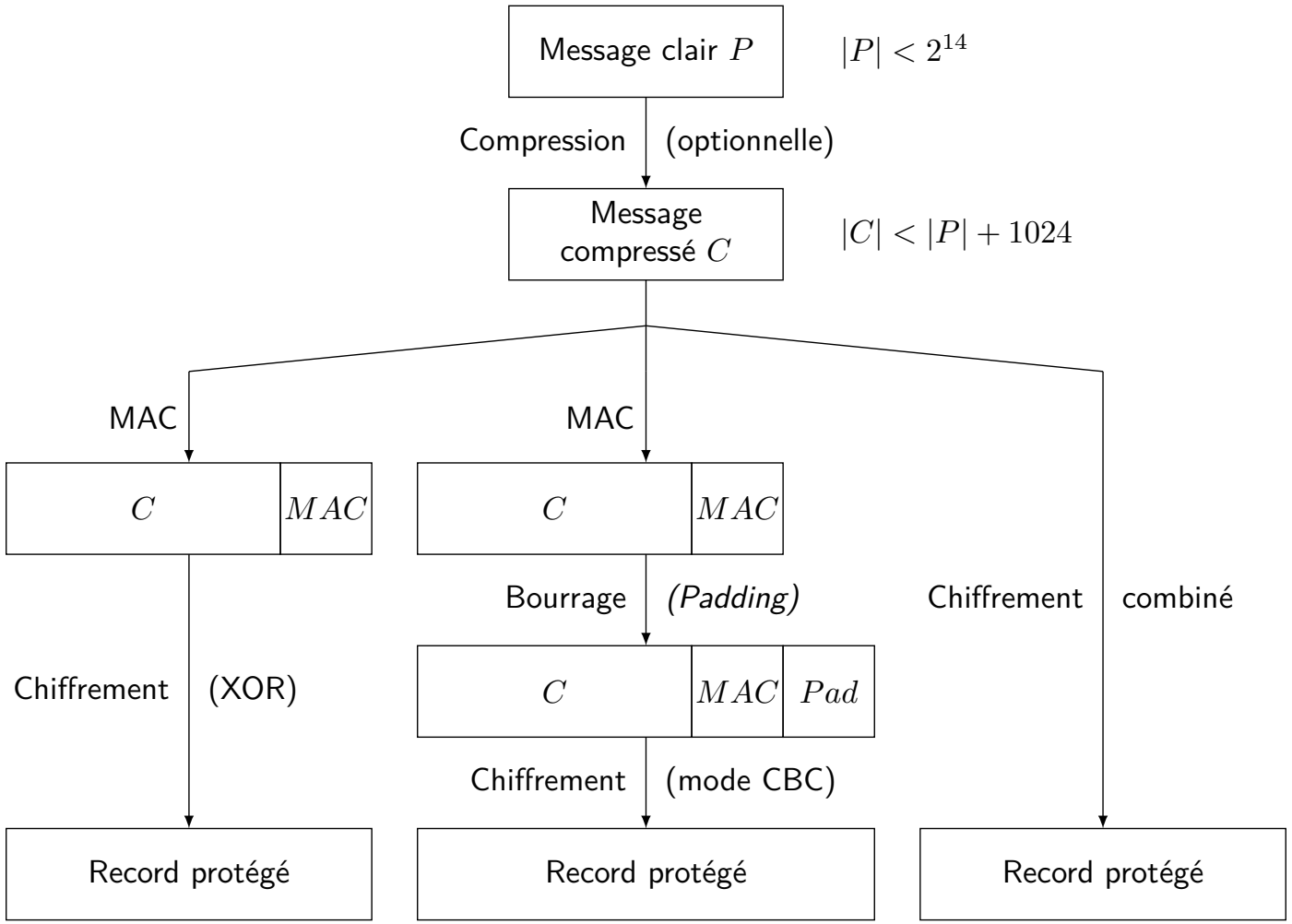
- SSLv2 n'incluait que les aléas dans le message `Finished`
- SSLv3 et TLS répètent la même erreur dans la dérivation de clé et la signature
- cette couverture incomplète rend possibles des attaques par confusion possibles (LogJam)

Solution : *Hash all the things !*

- [TLS 1.3] dans l'authentification par signature explicite
- [TLS 1.3] dans la dérivation de clé
- [TLS 1.3] dans le message `Finished`

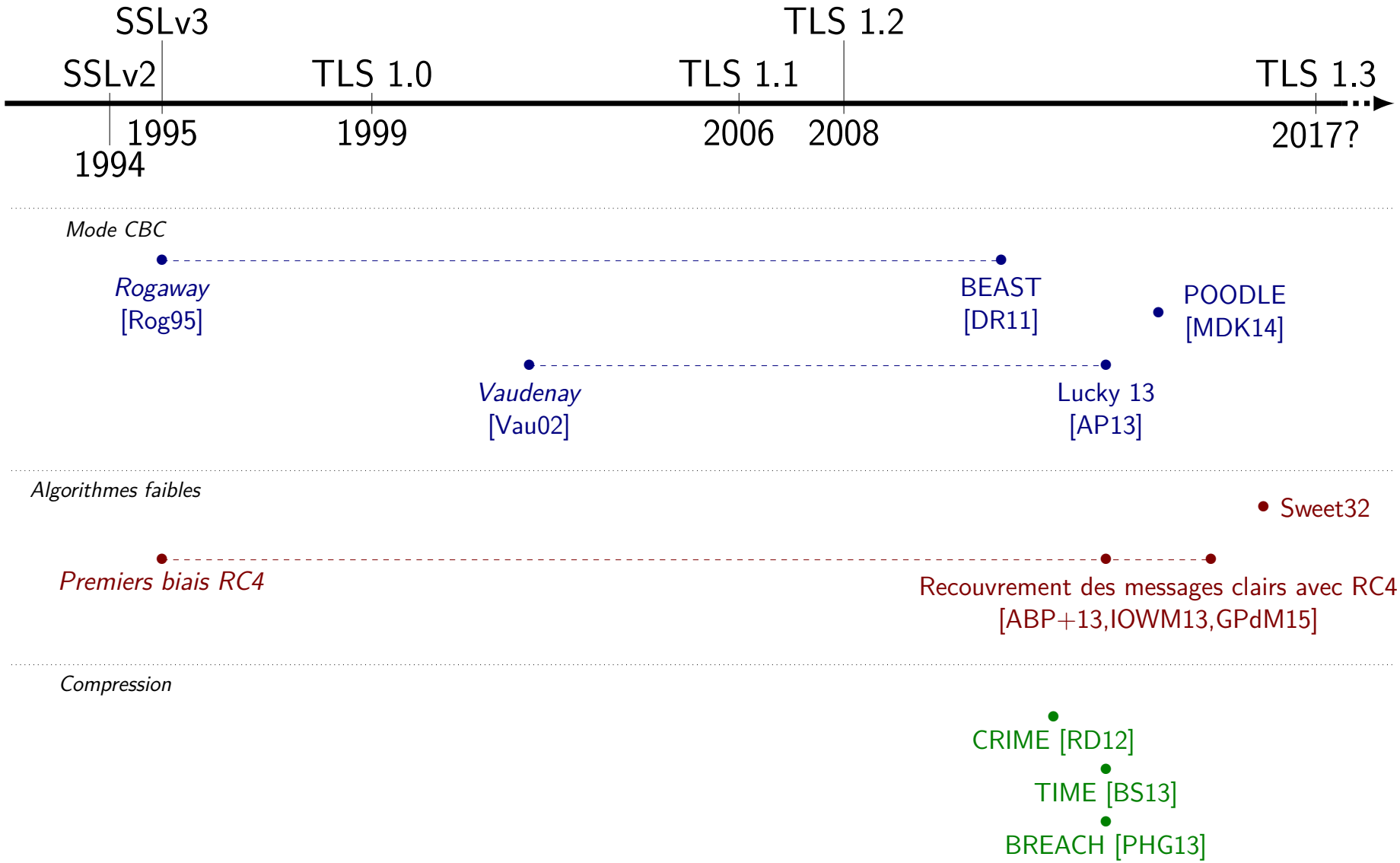
Failles dans le *Record Protocol*

Description du *Record Protocol*



Chiffrement par flot *Chiffrement par bloc en mode CBC* *Chiffrement combiné (AEAD)*

Attaques contre le Record Protocol



Réflexions autour du mode CBC

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

Bodo Möller

Le document indique que

Réflexions autour du mode CBC

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

Bodo Möller

Le document indique que

- 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

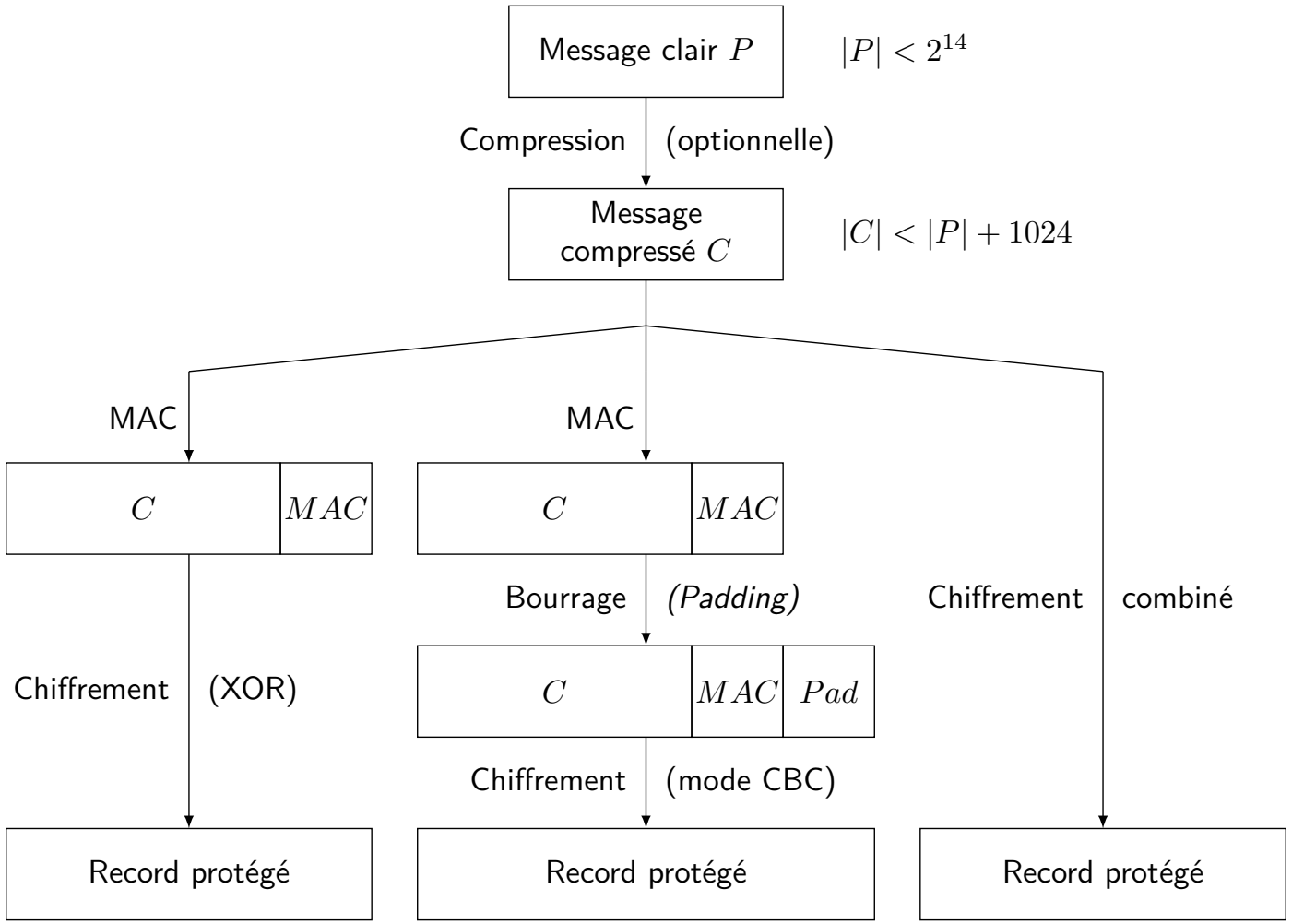
`https://www.openssl.org/~bodo/tls-cbc.txt`

Bodo Möller (2004-05-20)

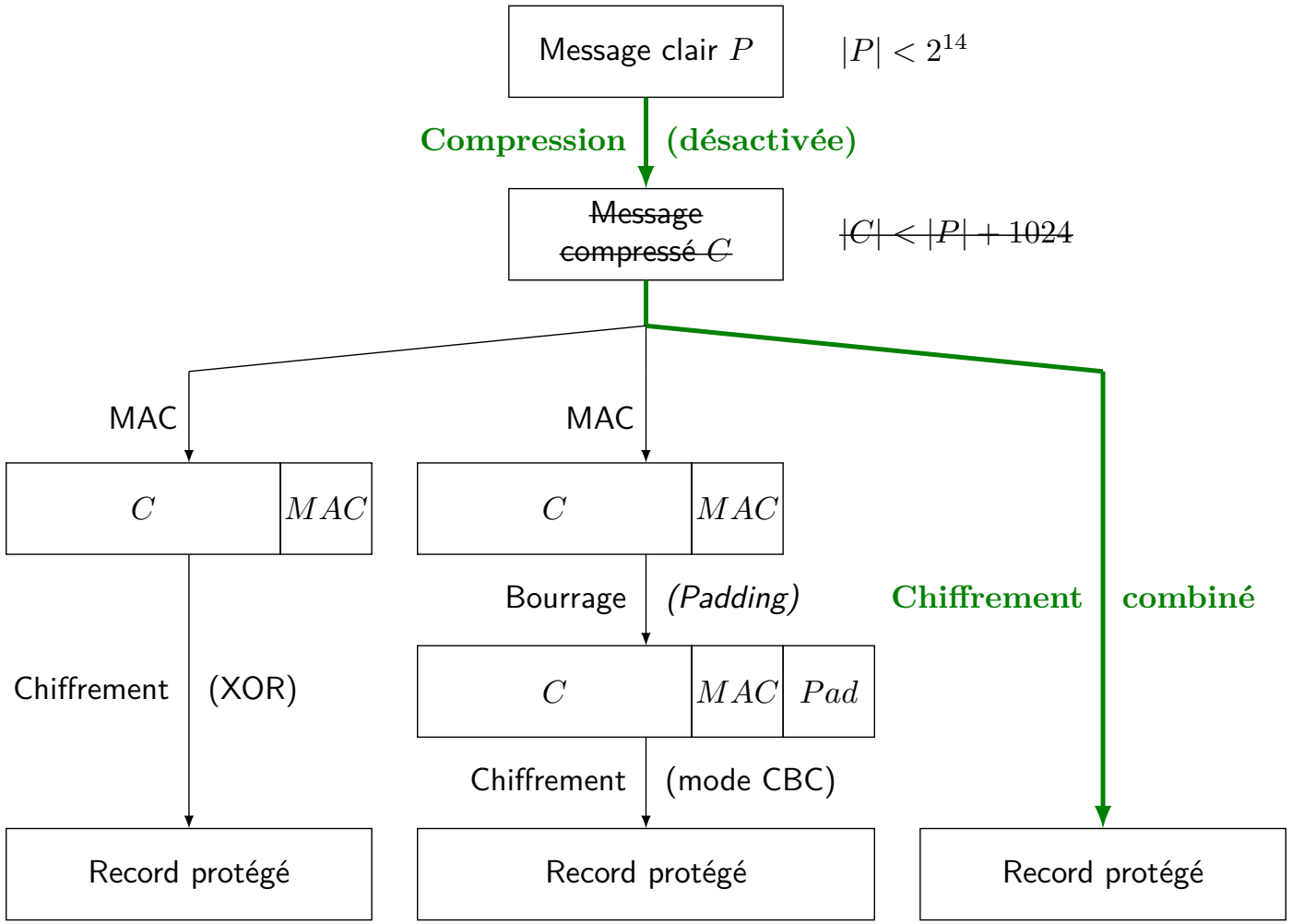
Le document indique que

- 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 long-terme



Record Protocol : la solution long-terme [TLS 1.3]



Erreurs d'implémentation

2014 : une année dure pour TLS

Toutes les piles TLS majeures touchées par une vulnérabilité critique

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

2014 : une année dure pour TLS

Toutes les piles TLS majeures touchées par une vulnérabilité critique

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

Quizz

- Que s'est-il aussi passé début avril 2014 (en plus de Heartbleed) ?
- Même question le 24 septembre 2014 (publication de Berserk) ?

Morceaux choisis (1/3) : True, False, `FILE_NOT_FOUND`

CVE-2014-0092 sur 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).

Morceaux choisis (2/3) : L'oubli de l'extension `Basic constraints`

Un *bug* de Microsoft Internet Explorer découvert par Marlinspike en 2002 :

- la pile X.509 ne vérifiait pas l'extension `Basic Constraints`
- tout certificat final pouvait servir de certificat d'autorité

Le même *bug*, en 2010, dans Apple iOS...

Morceaux choisis (2/3) : L'oubli de l'extension `Basic constraints`

Un *bug* de Microsoft Internet Explorer découvert par Marlinspike en 2002 :

- la pile X.509 ne vérifiait pas l'extension `Basic Constraints`
- tout certificat final pouvait servir de certificat d'autorité

Le même *bug*, en 2010, dans Apple iOS...

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

Morceaux choisis (3/3) : TLS dans tous ses états

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)
- 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

Réflexions sur les failles d'implémentation

Attention à ne pas blâmer trop vite le développeur

- implémenter certaines primitives crypto proprement relève du jonglage
- au vu des erreurs répétées, les spécifications sont trop complexes ?
- où étaient les tests *négatifs* de base ?

Éléments de solution

- [TLS 1.3 ?] spécifications claires
- le retrait des primitives obsolètes
  - [TLS 1.3] de la spécification
  - des déploiements
  - et *in fine* du code !
- de meilleures méthodologies de développements
  - utilisation de langages sûrs ou d'outils d'analyse de code
  - plus de tests

TLS 1.3 : un nouvel espoir ?

TLS 1.3 : un travail initié il y a plus de 3 ans

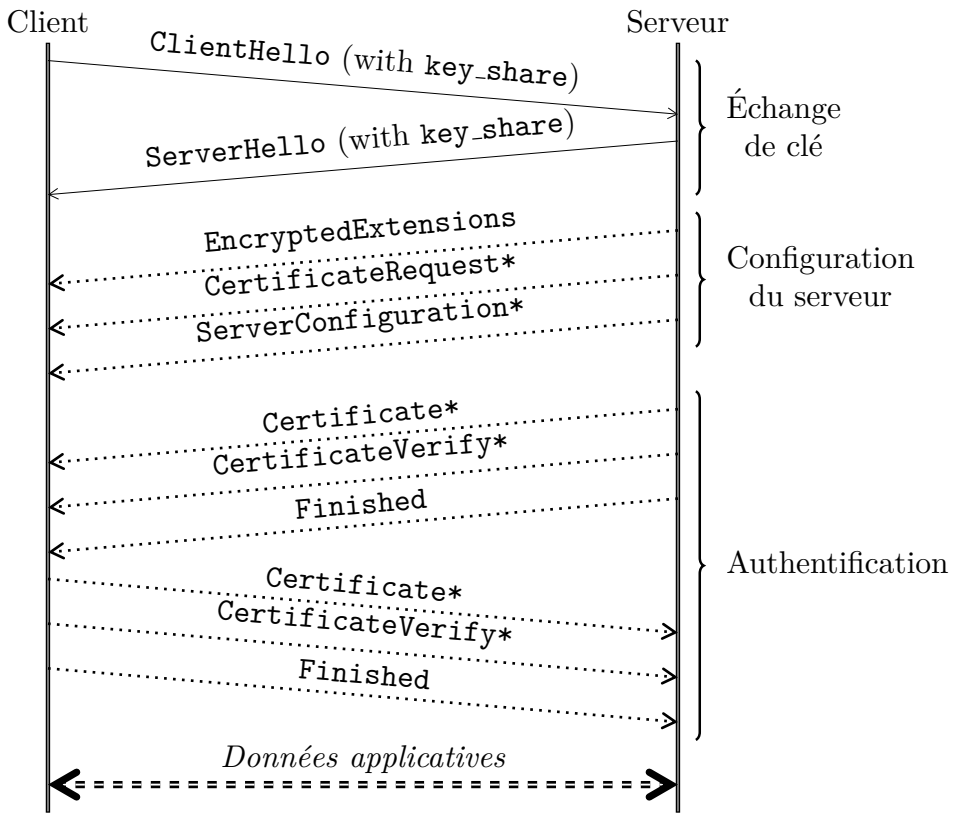
Rapide historique de TLS 1.3

- novembre 2013 : premiers éléments sur la liste IETF
- avril 2014 : TLS 1.3 *draft* 00
- janvier 2017 : *Working Group Last Call*
- avril 2017 : *draft* 20

Cahier des charges

- dépoussiérer le protocole
- accélérer l'établissement de sesion
- prendre en compte les vulnérabilités publiées

TLS 1.3 : plaquette commerciale



TLS 1.3 : un protocole...

- avec de la cryptographie moderne
- plus rapide
- plus simple*
- plus sûr*

Le côté obscur de TLS 1.3

Authentification tardive du client

- le serveur peut demander un certificat au client à tout moment
- il peut envoyer plusieurs requêtes en parallèle
- le client doit maintenir un état non négligeable
- *fonctionnalité optionnelle depuis début 2017*

Le côté obscur de TLS 1.3

Authentification tardive du client

- le serveur peut demander un certificat au client à tout moment
- il peut envoyer plusieurs requêtes en parallèle
- le client doit maintenir un état non négligeable
- *fonctionnalité optionnelle depuis début 2017*

Mode 0 RTT

- utilisation de la PSK pour chiffre des données dès le `ClientHello`
- rejeu de messages possibles
- question de la *forward secrecy*
- complexification de l'automate
- *cette fonctionnalité est optionnelle*

TLS 1.3 : une transition longue et difficile à prévoir

État des lieux

- TLS 1.3 devrait être publié en 2017
- quelques piles serveurs (Google/Cloudflare)
- quelques piles client (NSS/Chrome)
- le support TLS 1.3 est dans la branche de développement d'OpenSSL

Transition

- il faudra cohabiter avec les versions précédentes de TLS encore au moins une décennie
- SSLv3 vient juste de disparaître du monde HTTP
- mais SSLv2 est encore d'actualité dans l'écosystème SMTP...

Conclusion

SSL/TLS est un protocole (trop) riche à l'histoire semée de vulnérabilités

Conclusion

SSL/TLS est un protocole (trop) riche à l'histoire semée de vulnérabilités

TLS 1.3 prend en compte de nombreux travaux de recherche

- prise en compte des avancées en cryptographie d'il y a 10-15 ans
- modélisation et preuve des spécifications
- dérivation d'implémentation sûres

Conclusion

SSL/TLS est un protocole (trop) riche à l'histoire semée de vulnérabilités

TLS 1.3 prend en compte de nombreux travaux de recherche

- prise en compte des avancées en cryptographie d'il y a 10-15 ans
- modélisation et preuve des spécifications
- dérivation d'implémentation sûres

Sécuriser les communications est un problème global

- au-delà de la spécification, il faut vérifier les implémentations existantes
- quid de la cohabitation avec SSLv2 – TLS 1.2 ?
- quid de la gestion de la confiance ?
- il faut enfin étudier les interactions avec la couche applicative

Questions ?

Merci pour votre attention

Pour plus d'information :
`https://www.ssi.gouv.fr/publication/une-etude-de-lecosysteme-tls/`
`http://paperstreet.picty.org/~yeye`

Compléments

TLS 1.3 : un protocole avec de la cryptographie moderne

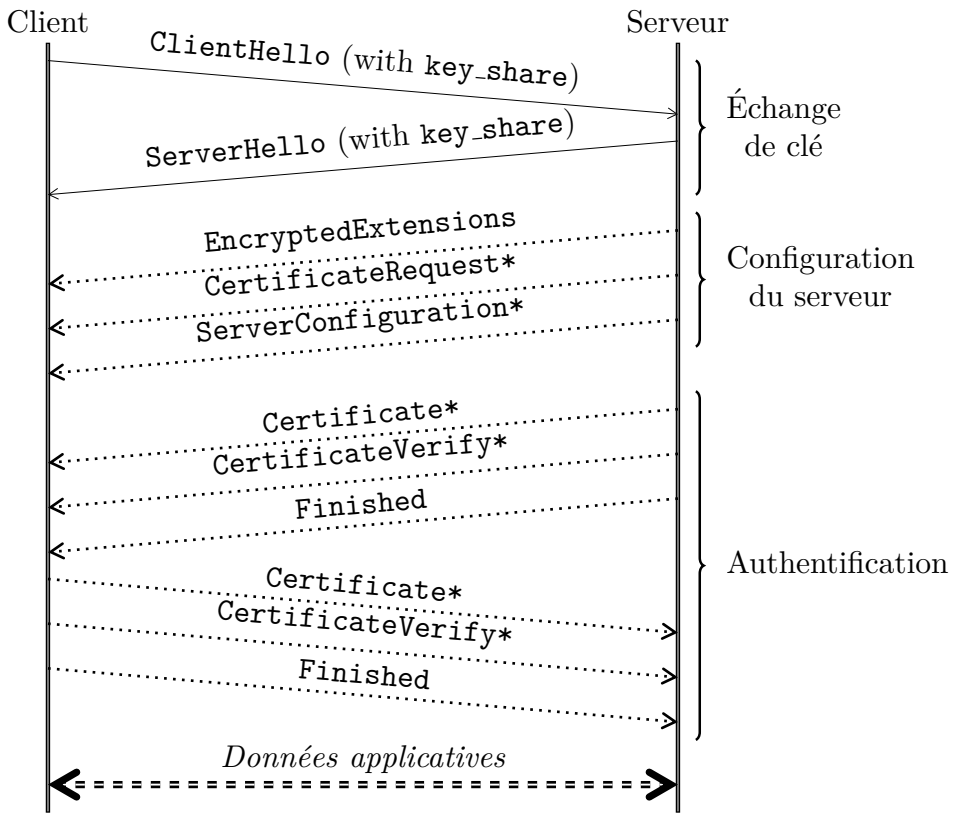
Handshake Protocol

- retrait de l'échange de clé par chiffrement RSA
- définition de groupes nommés pour (EC)DHE
- retrait de la renégociation
- signature de l'ensemble du transcript par le serveur
- mécanisme *anti-rollback*
- décorrellation du secret de reprise de session

Record Protocol

- retrait de la compression
- retrait du mode CBC
- retrait des algorithmes faibles (RC4, DES, 3DES)
- seul le mode AEAD est conservé (AES-GCM, ChaCha20+Poly1035)

TLS 1.3 : un protocole plus rapide



- 1 RTT si le client propose un groupe (EC)DHE connu du serveur
- 2 RTT sinon
- 0 RTT possible en cas de reprise de session

TLS 1.3 : un protocole plus simple et plus sûr

Beaucoup moins de fonctionnalités et d'options

- deux mode d'échange de clé : (EC)DHE et PSK *(Pre-Shared Key)*
- la reprise de session se fait par PSK
- un seul mode de protection des *records* : AEAD
- machine à état simplifiée

Depuis plus d'un an, de nombreux de travaux de vérification formelle ont été publiés

- janvier 2016 : atelier TRON en marge de NDSS
- avril 2017 : atelier TLS :DIV en marge d'Eurocrypt et EuroSSP
- miTLS (Inria Prosecco)
- modélisation utilisant Tamarin
- attaques cross-protocoles (TLS 1.3 et TLS 1.2/QUIC)

Les piles SSL/TLS maison

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

Les piles SSL/TLS maison

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

A `AES128-SHA`

Les piles SSL/TLS maison

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`

Les piles SSL/TLS maison

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

Les piles SSL/TLS maison

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`)

Les piles SSL/TLS maison

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

- 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 ?

Pourquoi *{False,Snap} Start* a échoué ?

Un feuilleton qui a duré plusieurs années

- en 2010, Google propose des extensions, *False Start* et *Snap Start*
- constat : une intolérance trop importante dans la nature
- abandon en 2012 de ces propositions

Pourquoi *{False,Snap} Start* a échoué ?

Un feuilleton qui a duré plusieurs années

- en 2010, Google propose des extensions, *False Start* et *Snap Start*
- constat : une intolérance trop importante dans la nature
- abandon en 2012 de ces propositions

- un an plus tard, le problème resurgit dans un autre contexte
- on apprend sur la liste `tls@ietf.org` qu'en fait, le problème vient d'un `ClientHello` trop gros...

Ce paquet est une chimère

Analysons les premiers octets d'un `ClientHello` de 258 octets

16	03	01	01	02

Ce paquet est une chimère

Analysons les premiers octets d'un `ClientHello` de 258 octets

16	03	01	01	02

Type	Version	Longueur
HS	*TLS 1.0*	*258*

TLS

Ce paquet est une chimère

Analysons les premiers octets d'un `ClientHello` de 258 octets

16	03	01	01	02

Type	Version	Longueur
HS	*TLS 1.0*	*258*

TLS

Longueur	*Pad.*	Type
5635	...	*CH*

SSLv2

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

Ce paquet est une chimère

Analysons les premiers octets d'un `ClientHello` de 258 octets

16	03	01	01	02

Type	Version	Longueur
HS	*TLS 1.0*	*258*

TLS

Longueur	*Pad.*	Type
5635	...	*CH*

SSLv2

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

Google a proposé une extension pour ajouter du bourrage...