SSL/TLS : état des lieux et recommandations

Olivier Levillain

ANSSI

4 juin 2012

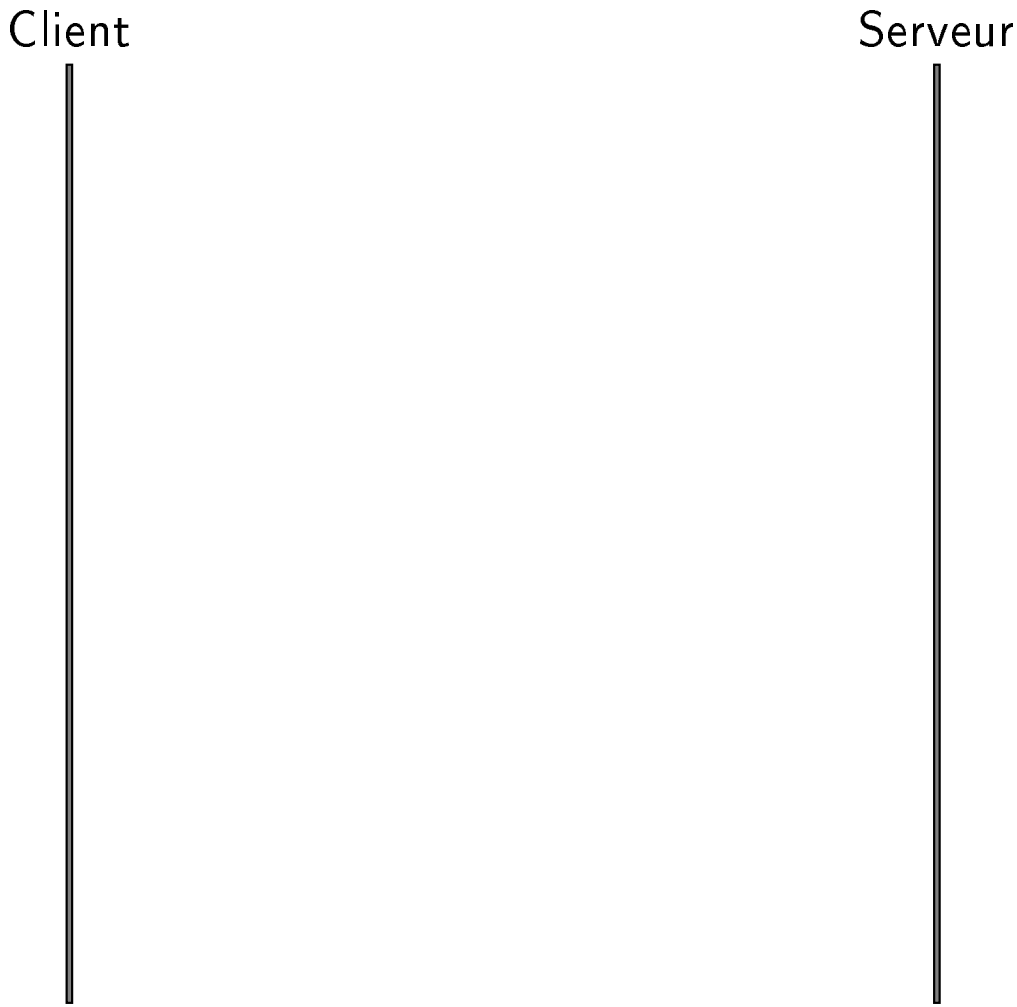
SSL/TLS : une brique essentielle d'Internet

- `https://` inventé par Netscape en 1995
  - début du commerce en ligne
  - première version très mal conçue
  - plusieurs successeurs potentiels dès 96 :
    - PCT (Microsoft)
    - SSLv3 (Netscape)
- Utilisation massive aujourd'hui
  - HTTPS, bien au-delà du commerce en ligne
  - Sécurisation d'autres protocoles (SMTP, IMAP, LDAP, etc.)
  - VPN SSL
  - EAP TLS

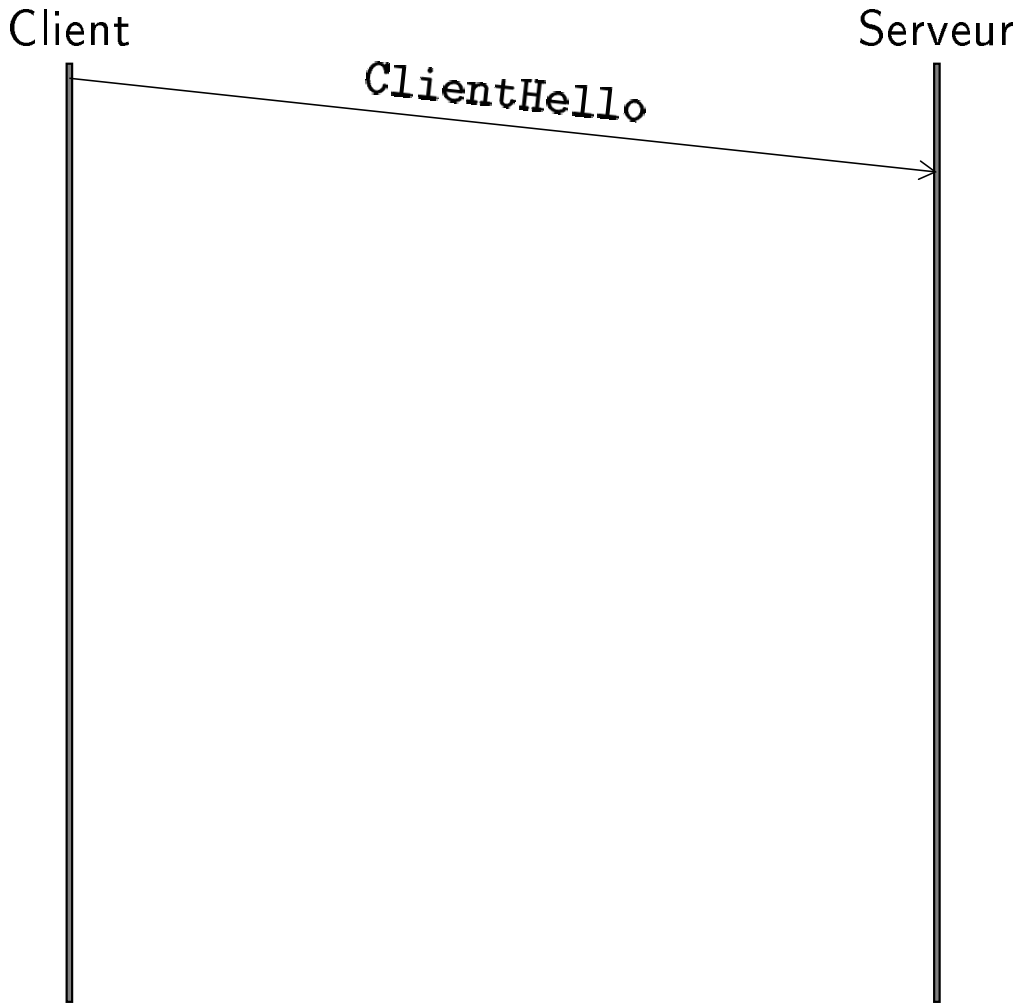
Un peu d'histoire

Version	Année	Observations
SSLv2	1995	Protocole conçu par Netscape Failles structurelles majeures
SSLv3	1996	Faille dans l'implémentation de RSA Problèmes d'interopérabilité
TLSv1.0	2001	Attaques crypto sur le mode CBC Des solutions de contournement existent
TLSv1.1	2006	Version minimale conseillée
TLSv1.2	2008	Support de nouveaux algorithmes

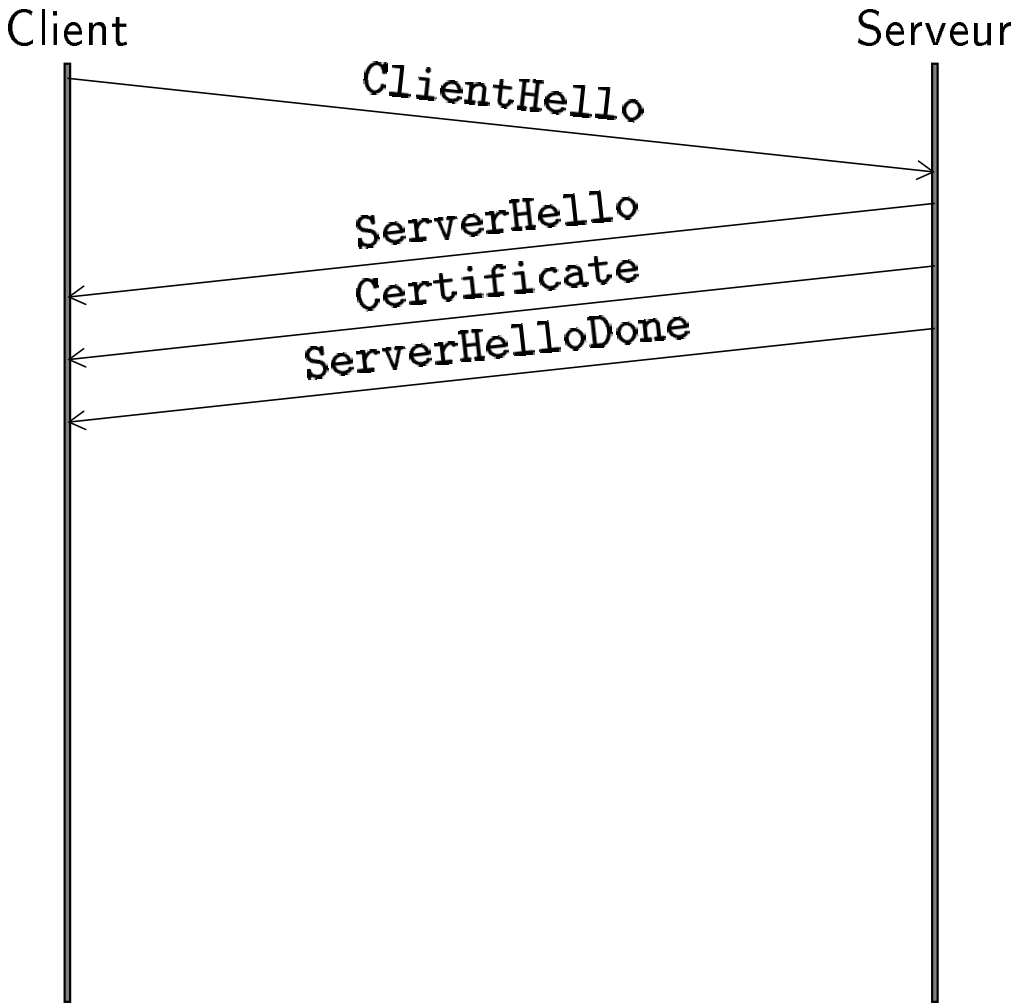
Fonctionnement du protocole



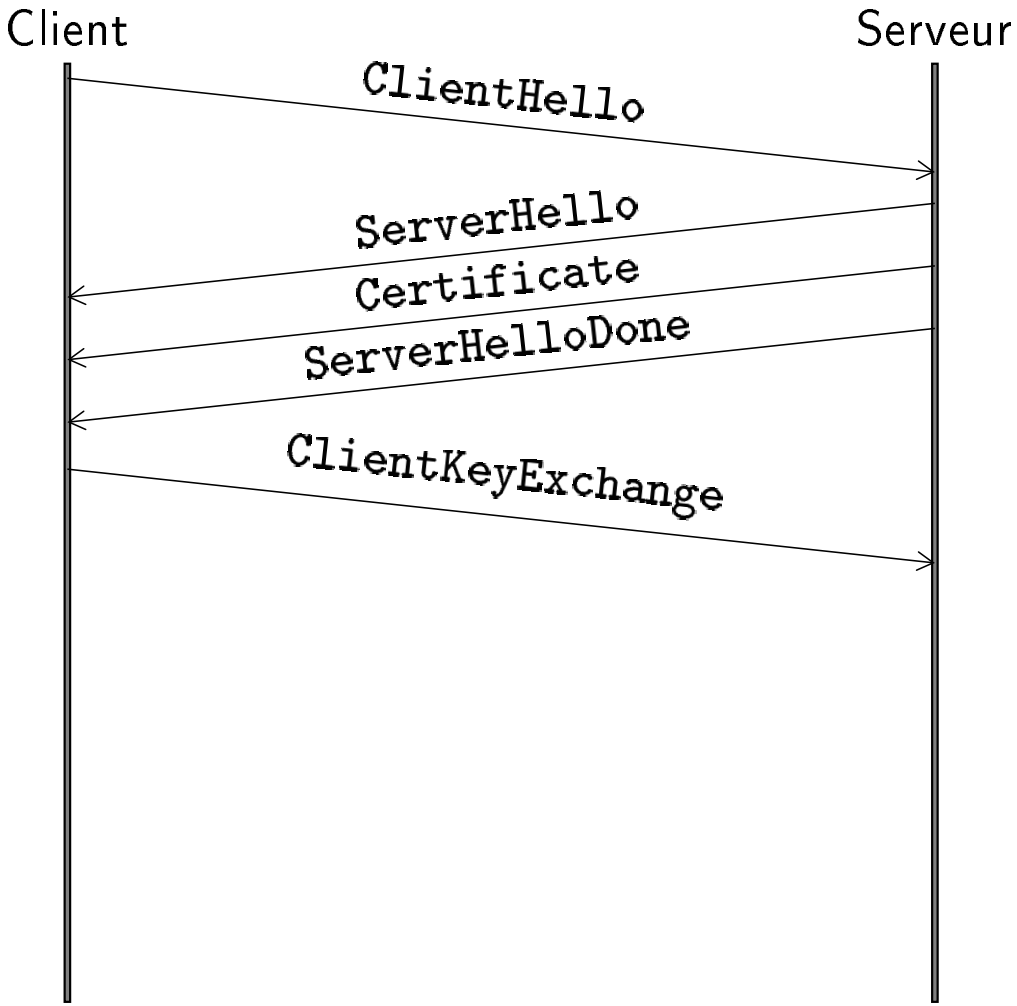
Fonctionnement du protocole



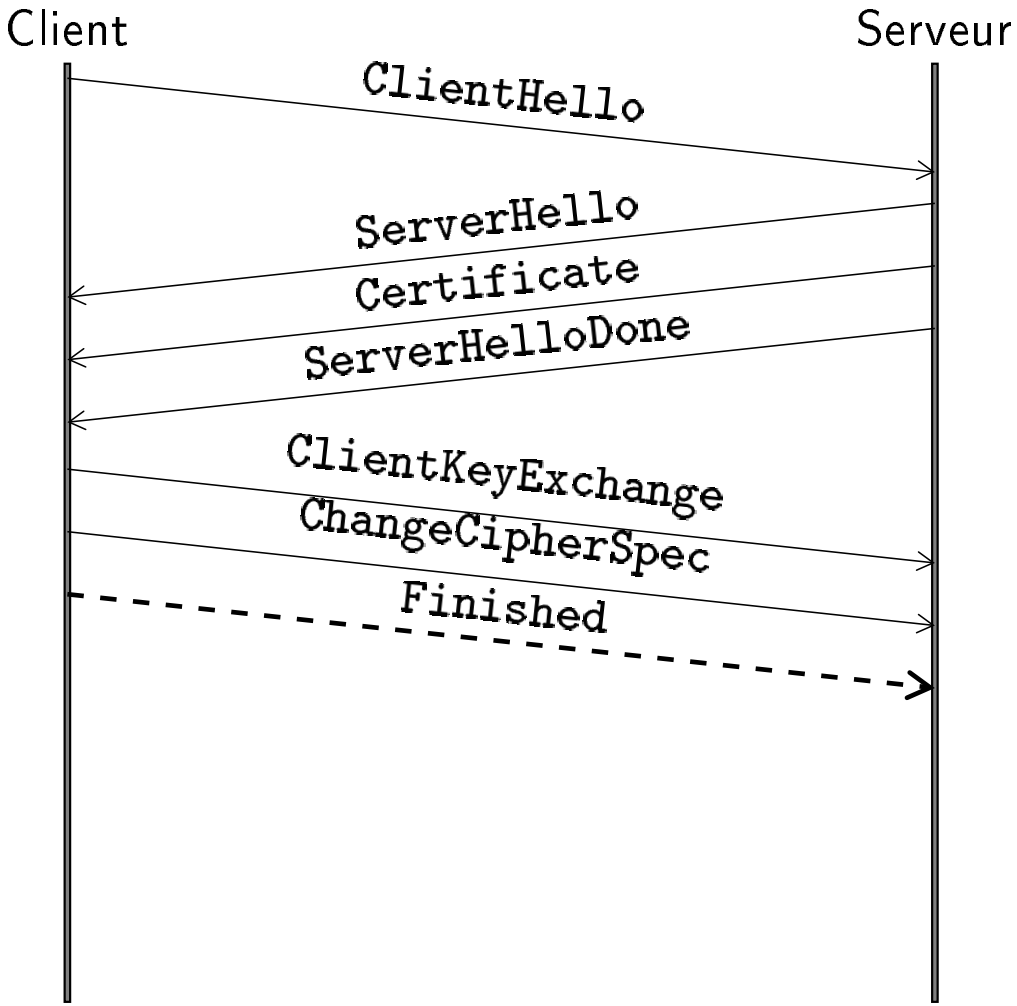
Fonctionnement du protocole



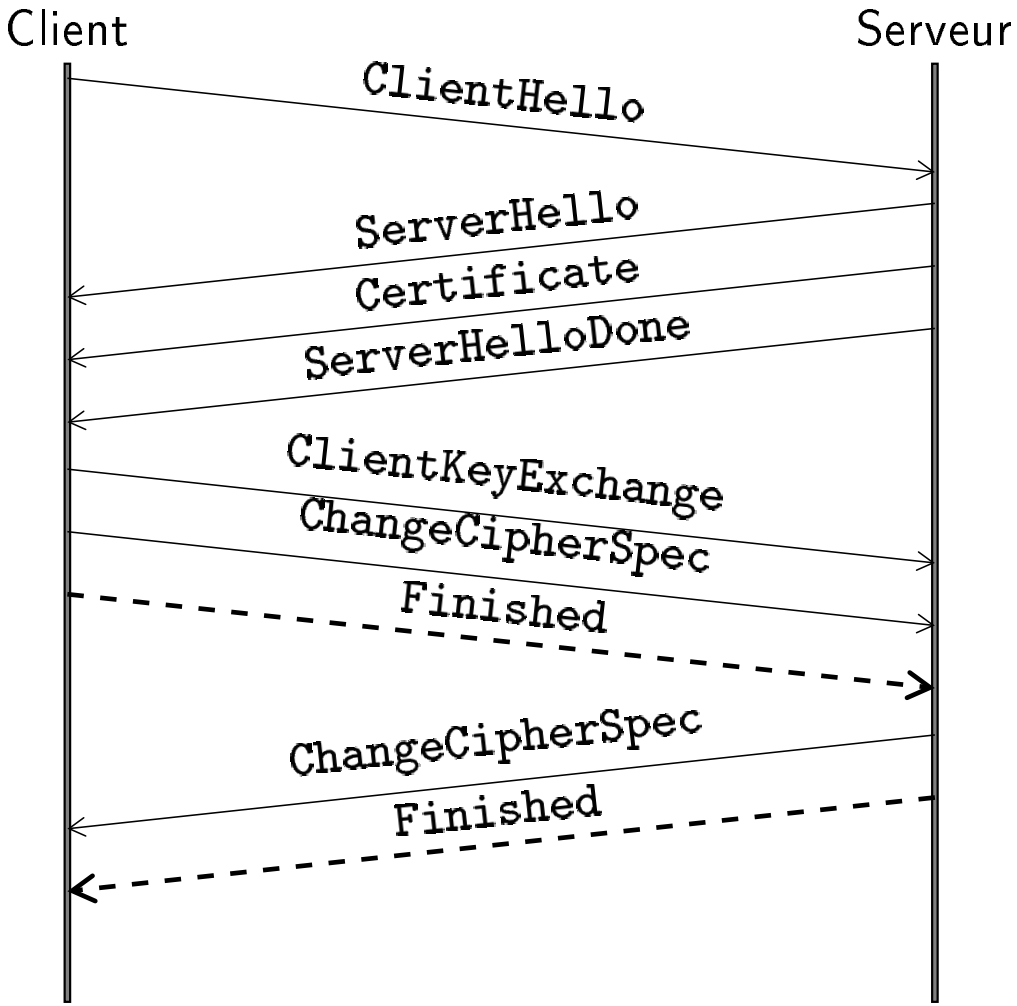
Fonctionnement du protocole



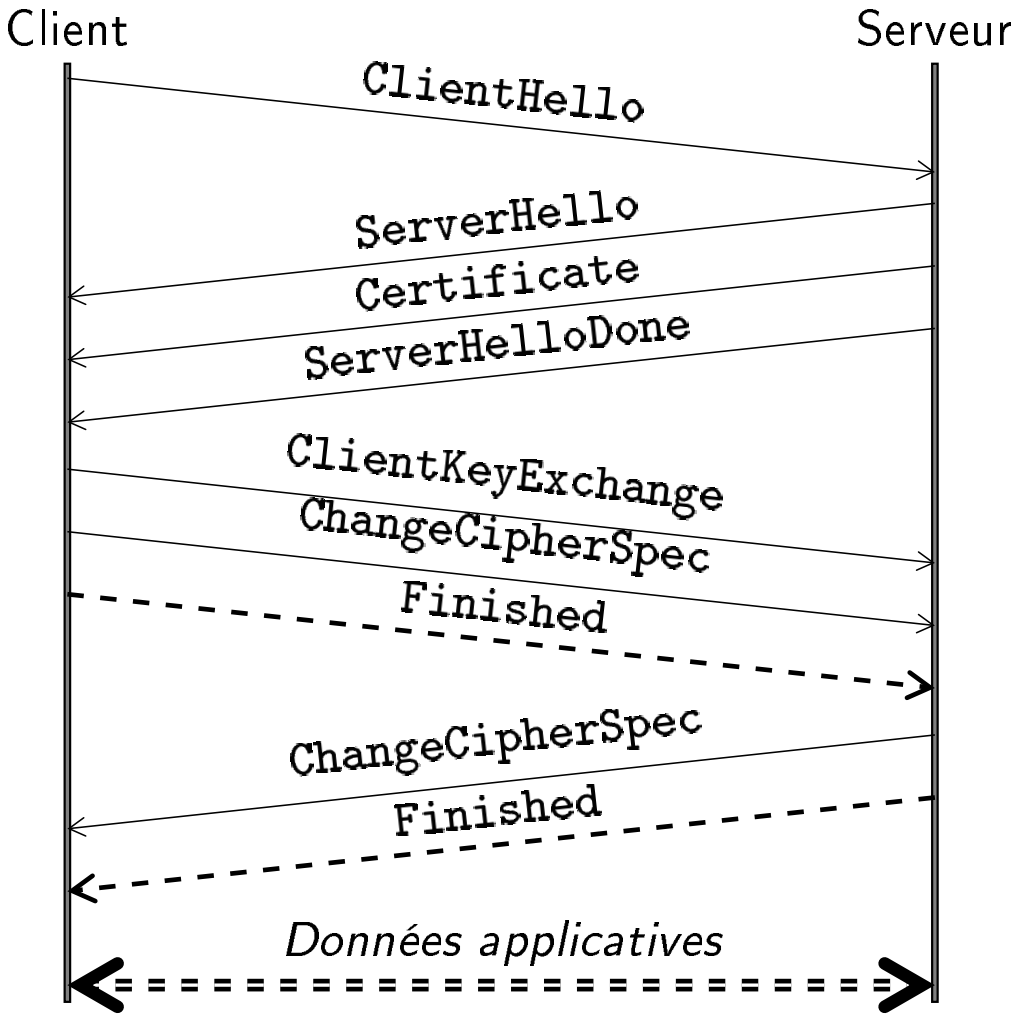
Fonctionnement du protocole



Fonctionnement du protocole



Fonctionnement du protocole



Implémentations connues

- OpenSSL
- GnuTLS
- NSS
- xyssl devenu polarssl
- Crypto API (jusqu'à Windows XP et Windows 2003)
- Crypto NG (Windows 7 et Windows 2008)
- implémentation Apple (MacOS + Safari)
- implémentation Opera
- et d'autres implémentations plus ou moins indépendantes
  - Java
  - Equipements réseau
  - ...

SSLv2 : une version à proscrire

- SSLv2 publié en 1995

SSLv2 : une version à proscrire

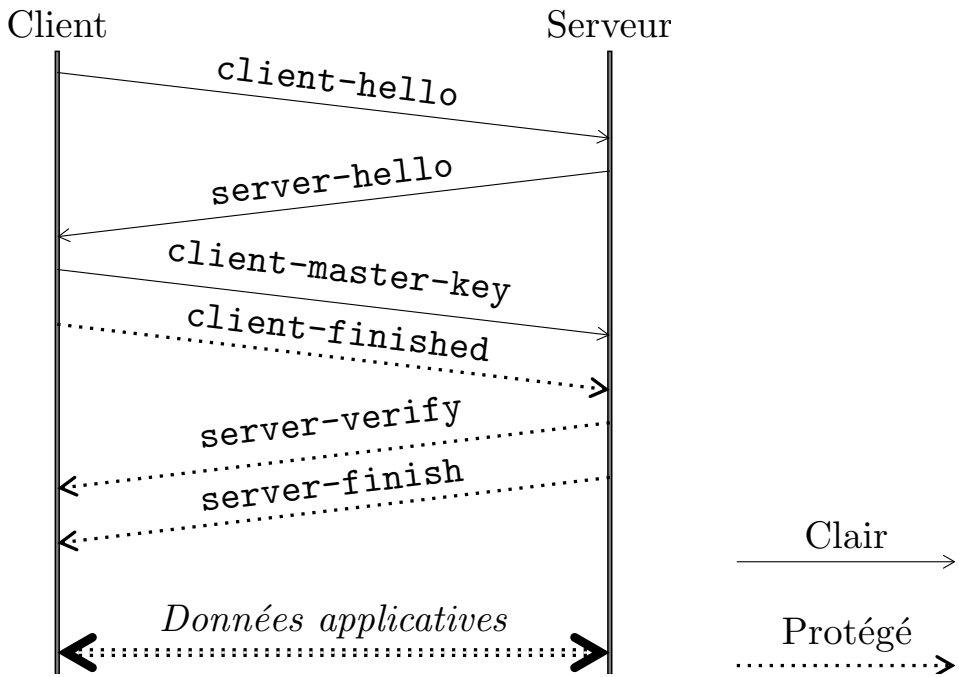
- SSLv2 publié en 1995
- Protocole considéré comme dangereux
- RFC 6176 en mars 2011 (*Prohibiting SSLv2*)
  - HMAC MD5 pour l'intégrité
  - partage de la clé pour l'intégrité et la confidentialité
  - mauvaise gestion de la fin de connexion (attaques par troncature du flux possibles)
  - possibilité de faire une négociation à la baisse des algorithmes

SSLv2 : une version à proscrire

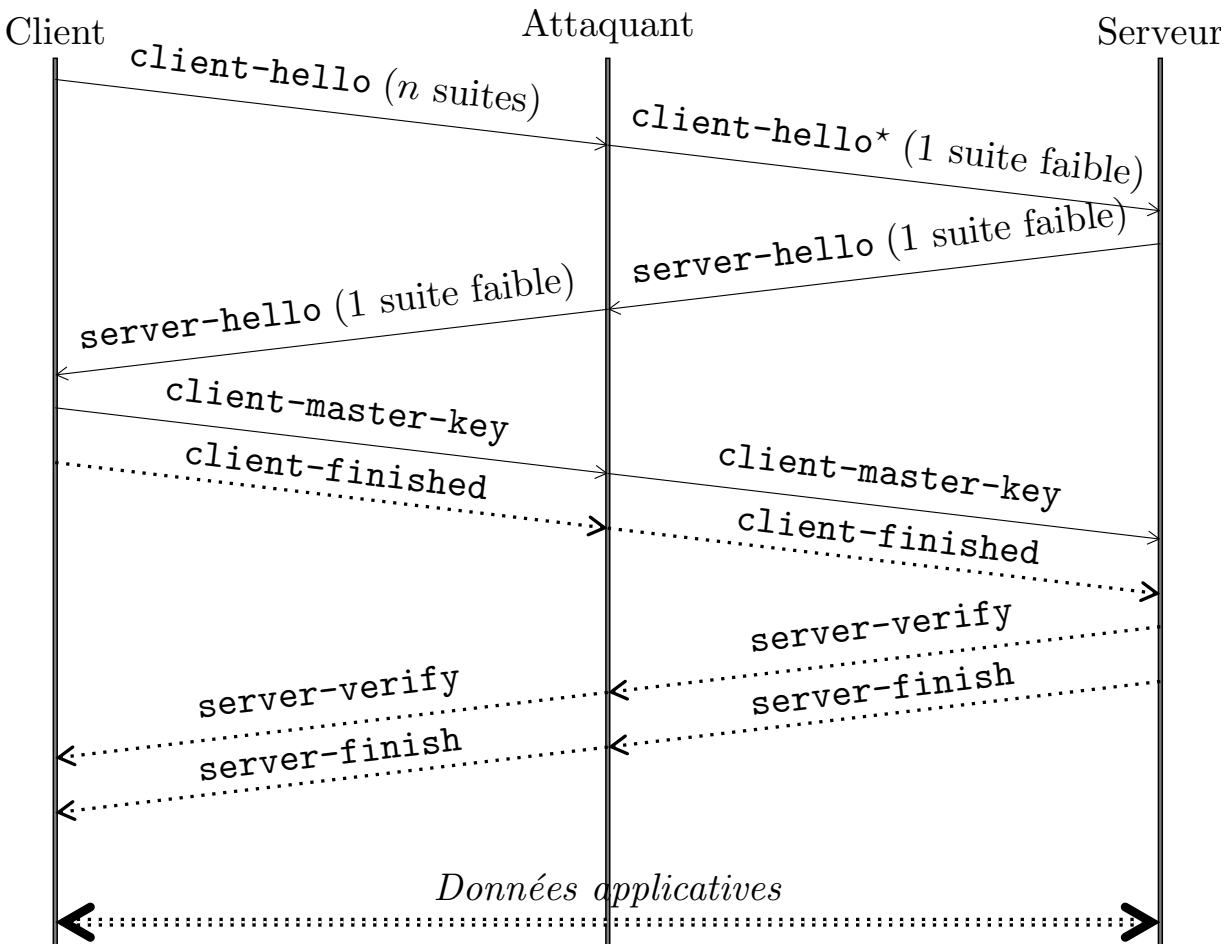
- SSLv2 publié en 1995
- Protocole considéré comme dangereux
- RFC 6176 en mars 2011 (*Prohibiting SSLv2*)
  - HMAC MD5 pour l'intégrité
  - partage de la clé pour l'intégrité et la confidentialité
  - mauvaise gestion de la fin de connexion (attaques par troncature du flux possibles)
  - possibilité de faire une négociation à la baisse des algorithmes

Une bonne nouvelle : SSLv2 tend (enfin) à disparaître.

SSLv2 : Présentation des messages



Négociation à la baisse



SSLv3 : une mise à jour majeure

- SSLv3 publié en 1996

SSLv3 : une mise à jour majeure

- SSLv3 publié en 1996
- Deux gros soucis avec de vieilles implémentations SSLv3 :
  - attaque de Bleichenbacher en 1998 sur PKCS#1

SSLv3 : une mise à jour majeure

- SSLv3 publié en 1996
- Deux gros soucis avec de vieilles implémentations SSLv3 :
  - attaque de Bleichenbacher en 1998 sur PKCS#1
  - incompatibilité avec les extensions TLS (RFC 3546)
    - courbes elliptiques
    - reprise de session sans état côté serveur
    - renégociation sécurisée

SSLv3 : une mise à jour majeure

- SSLv3 publié en 1996
- Deux gros soucis avec de vieilles implémentations SSLv3 :
  - attaque de Bleichenbacher en 1998 sur PKCS#1
  - incompatibilité avec les extensions TLS (RFC 3546)
    - courbes elliptiques
    - reprise de session sans état côté serveur
    - renégociation sécurisée
- Une implémentation corrigeant ces deux défauts sera en pratique compatible avec TLSv1.0
  - SSLv3 ne présente donc aucun intérêt aujourd'hui

TLSv1.0 : attaque sur le mode CBC (1/2)

- En 2001, le protocole est repris par l'IETF et devient TLSv1.0 sans changement fondamental

TLSv1.0 : attaque sur le mode CBC (1/2)

- En 2001, le protocole est repris par l'IETF et devient TLSv1.0 sans changement fondamental
- Attaque de Rogaway en 2002 sur le mode CBC avec IV implicite...

TLSv1.0 : attaque sur le mode CBC (1/2)

- En 2001, le protocole est repris par l'IETF et devient TLSv1.0 sans changement fondamental
- Attaque de Rogaway en 2002 sur le mode CBC avec IV implicite...
- médiatisée en 2011 par Duong et Rizzo (BEAST)

TLSv1.0 : attaque sur le mode CBC (1/2)

- En 2001, le protocole est repris par l'IETF et devient TLSv1.0 sans changement fondamental
- Attaque de Rogaway en 2002 sur le mode CBC avec IV implicite...
- médiatisée en 2011 par Duong et Rizzo (BEAST)
- Réactions diverses
  - RC4 en priorité (incompatible avec DHE-RSA)
  - bricolage pour randomiser les IVs (en éclatant les messages)
  - passer à TLSv1.1

TLSv1.0 : attaques sur le mode CBC (2/2)

TLSv1.0 : attaques sur le mode CBC (2/2)

- Attaque de Rogaway en 2002
  - avant TLSv1.1, utilisation d'un IV implicite (chaînage des messages)
  - hypothèses lourdes, considérées irréalistes :
    - attaque à clair choisi
    - attaque adaptative (accès en lecture au chiffré)
  - correction dans TLSv1.1 (RFC 4346) avec un IV explicite, mais peu implémenté jusqu'à maintenant...

TLSv1.0 : attaques sur CBC (2/2)

- 2011, Thai Duong et Juliano Rizzo ont présenté BEAST :
  - réutilisation d'un même canal TLS entre onglets
  - contournement de la *Same Origin Policy*
  - attaque pratique pour voler des *cookies*

TLSv1.0 : attaques sur CBC (2/2)

- 2011, Thai Duong et Juliano Rizzo ont présenté BEAST :
  - réutilisation d'un même canal TLS entre onglets
  - contournement de la *Same Origin Policy*
  - attaque pratique pour voler des *cookies*

- Réactions diverses
  - choisir RC4 en priorité
  - bricolage pour randomiser les IVs (en éclatant les messages)
  - passer à TLSv1.1 (OpenSSL 1.0.1)

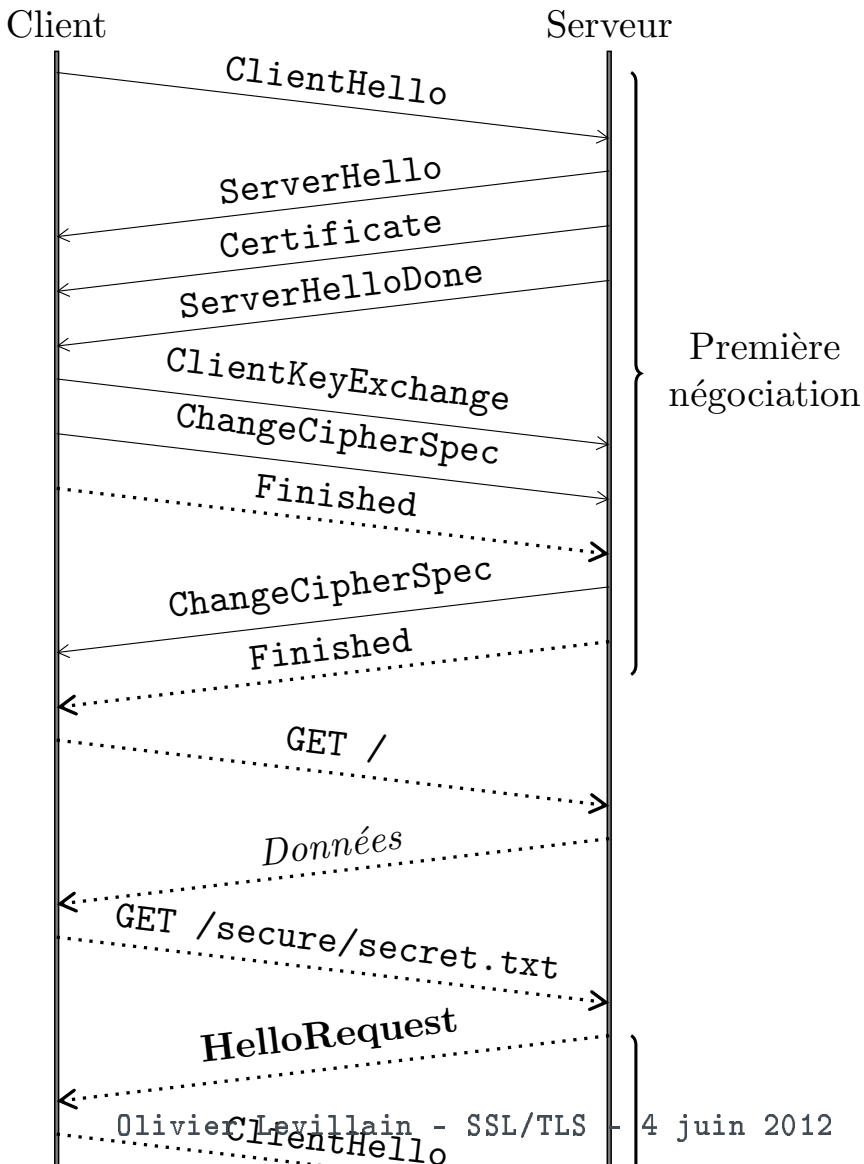
TLSv1.1 et 1.2 : en route vers le futur

- TLSv1.1 : version minimum conseillée

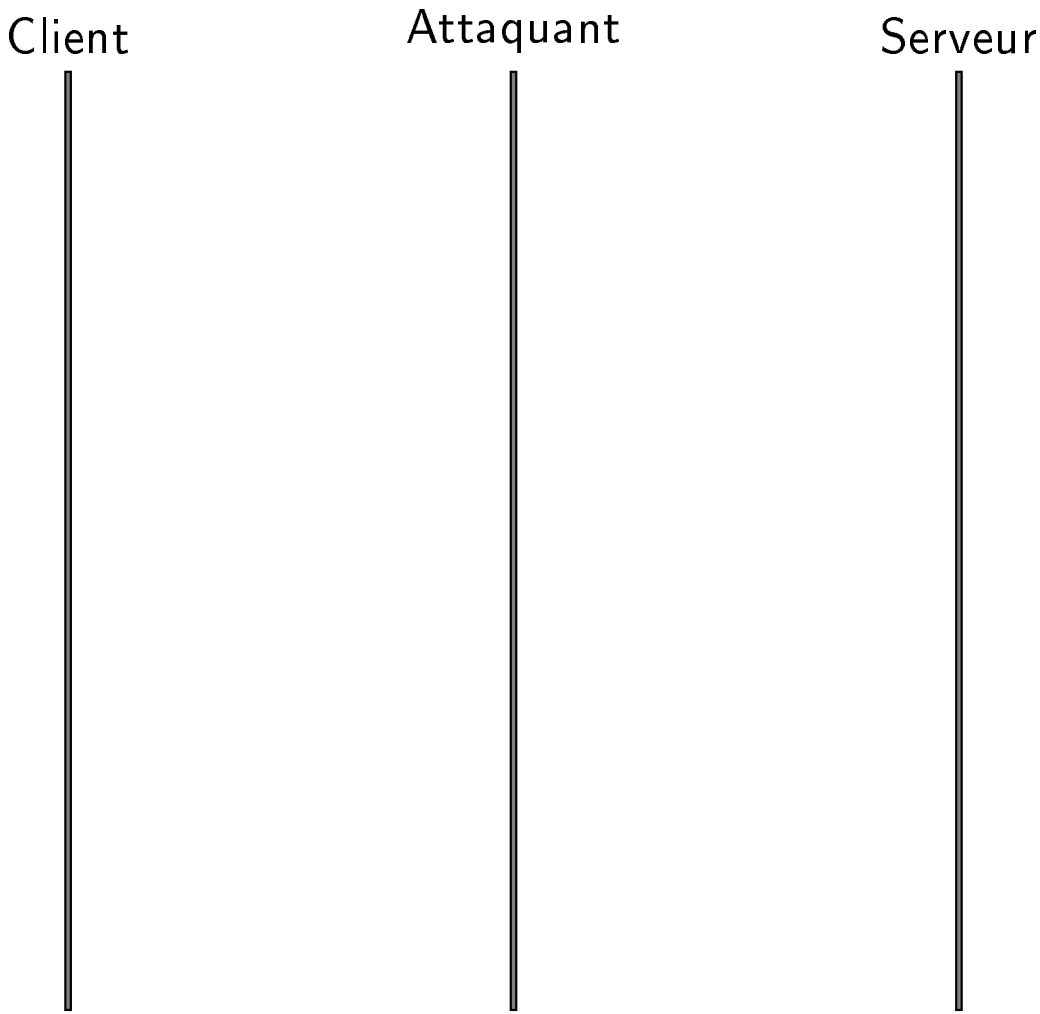
TLSv1.1 et 1.2 : en route vers le futur

- TLSv1.1 : version minimum conseillée
- TLSv1.2 est une refonte du standard
  - inclusion des extensions dans le standard
  - inclusion d'autres RFCs dans le cœur du standard
  - avec quelques petits cadeaux pour les algorithmes cryptographiques
    - modes combinés pour le chiffrement et l'intégrité (GCM)
    - ajout de suites cryptographiques avec HMAC SHA256
    - possiblité d'utiliser une PRF différente de MD5/SHA1

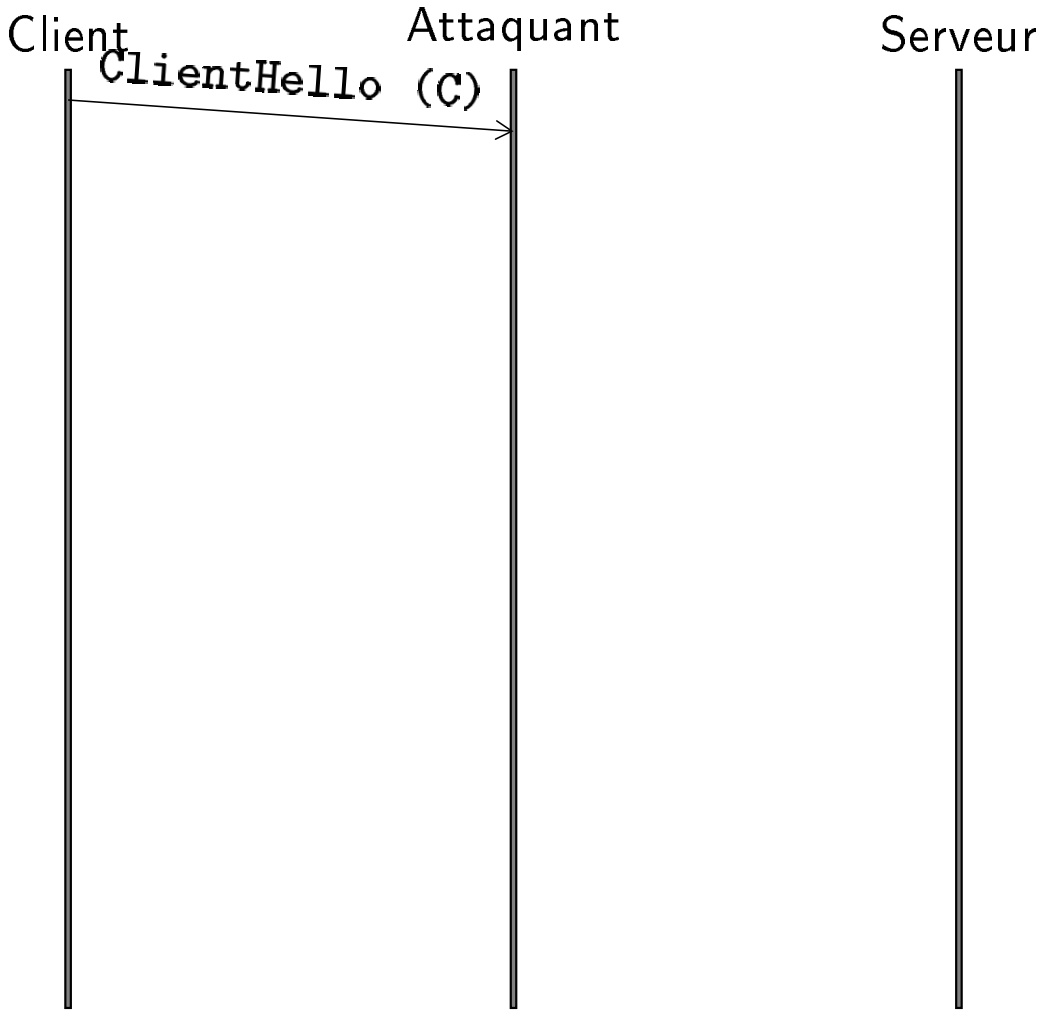
Description d'une renégociation légitime



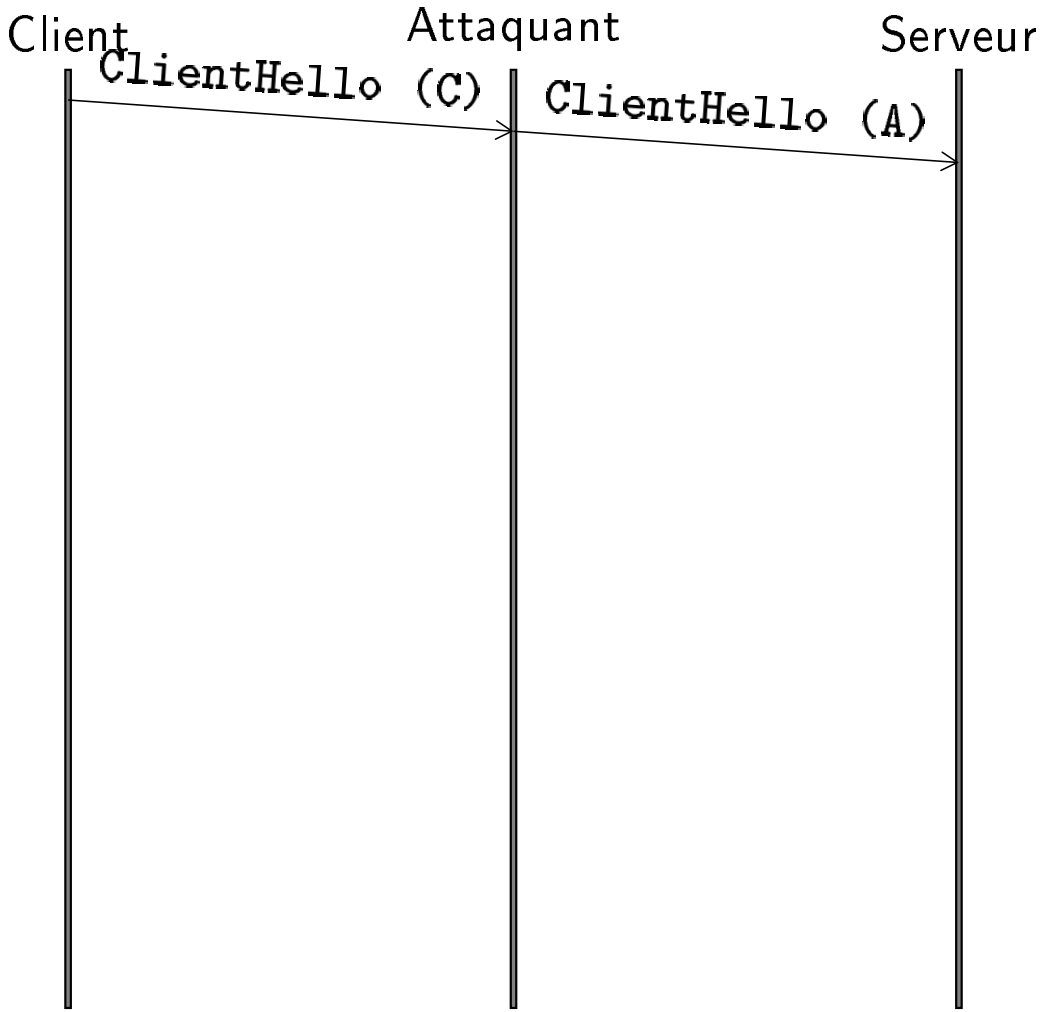
Vulnérabilité dans la renégociation



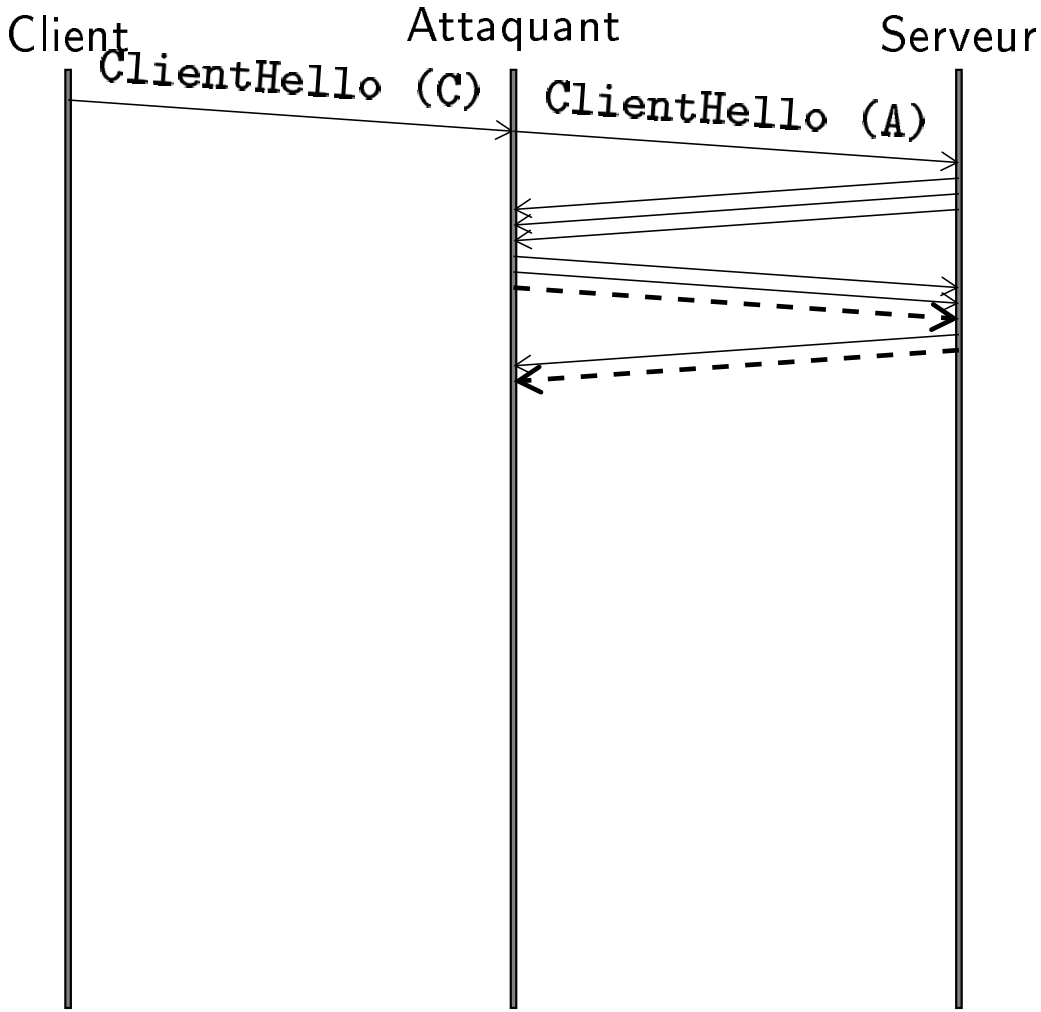
Vulnérabilité dans la renégociation



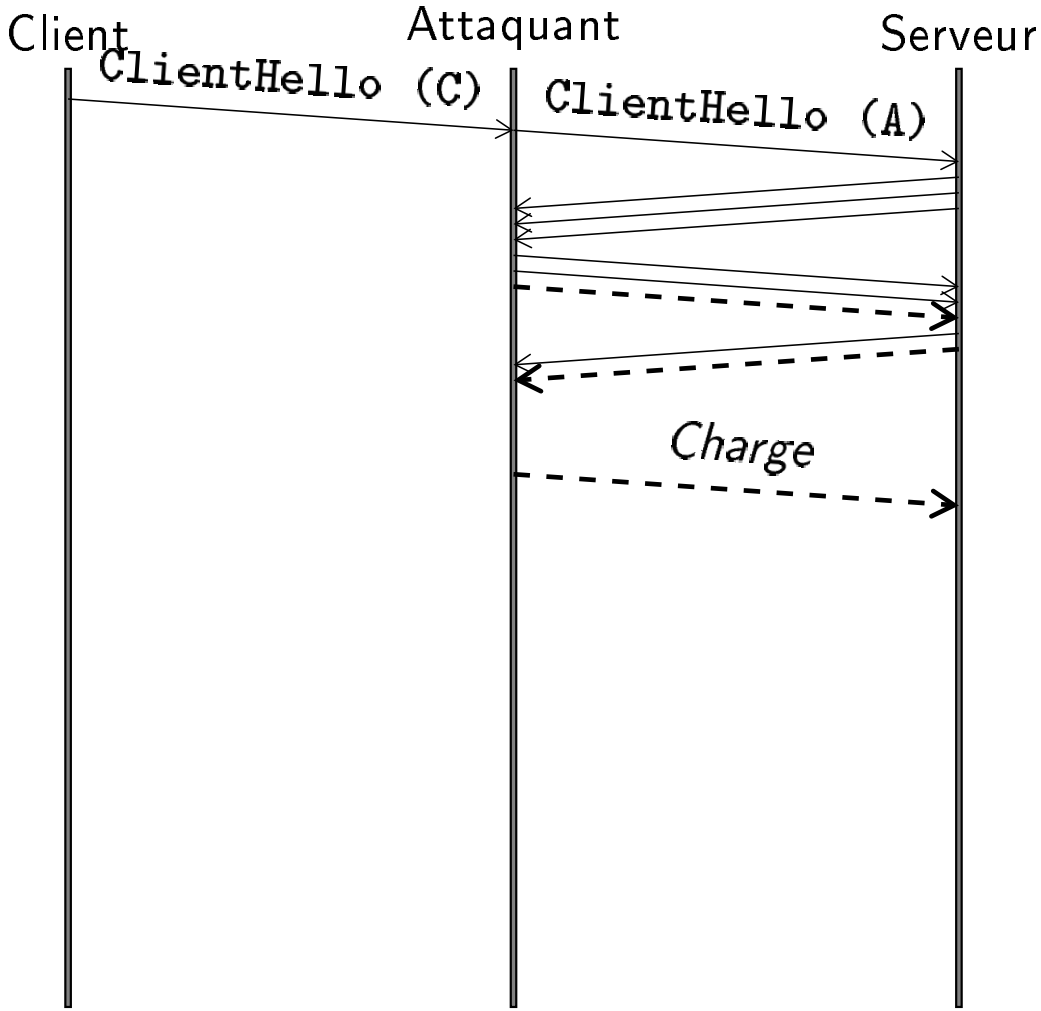
Vulnérabilité dans la renégociation



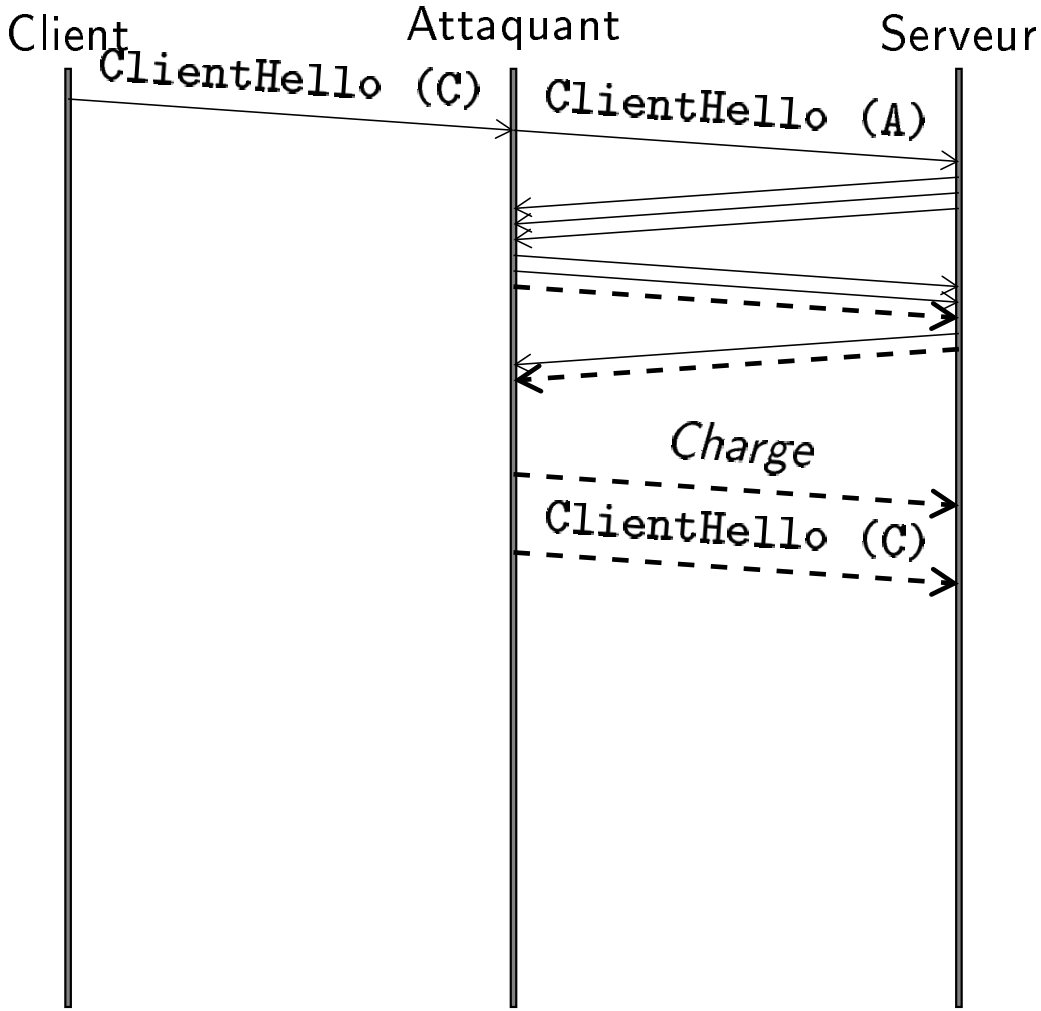
Vulnérabilité dans la renégociation



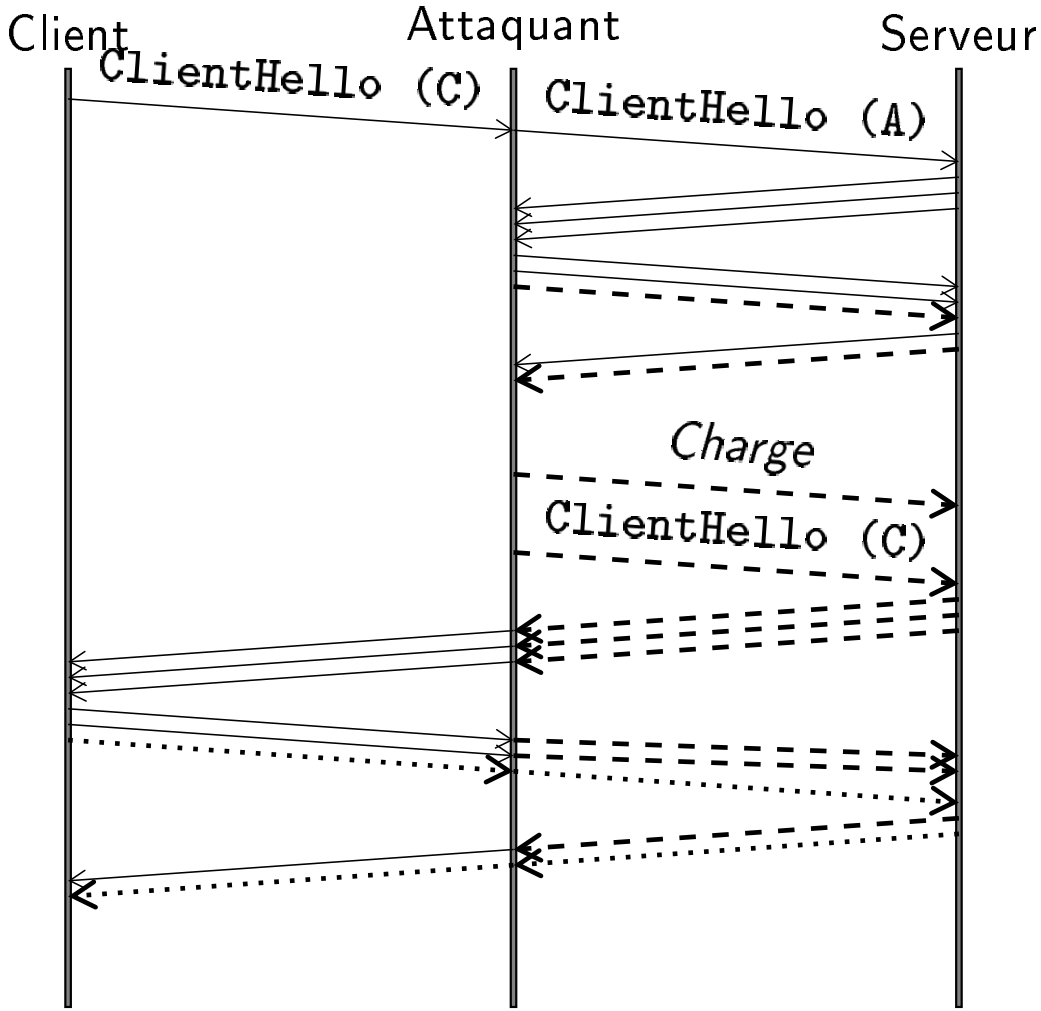
Vulnérabilité dans la renégociation



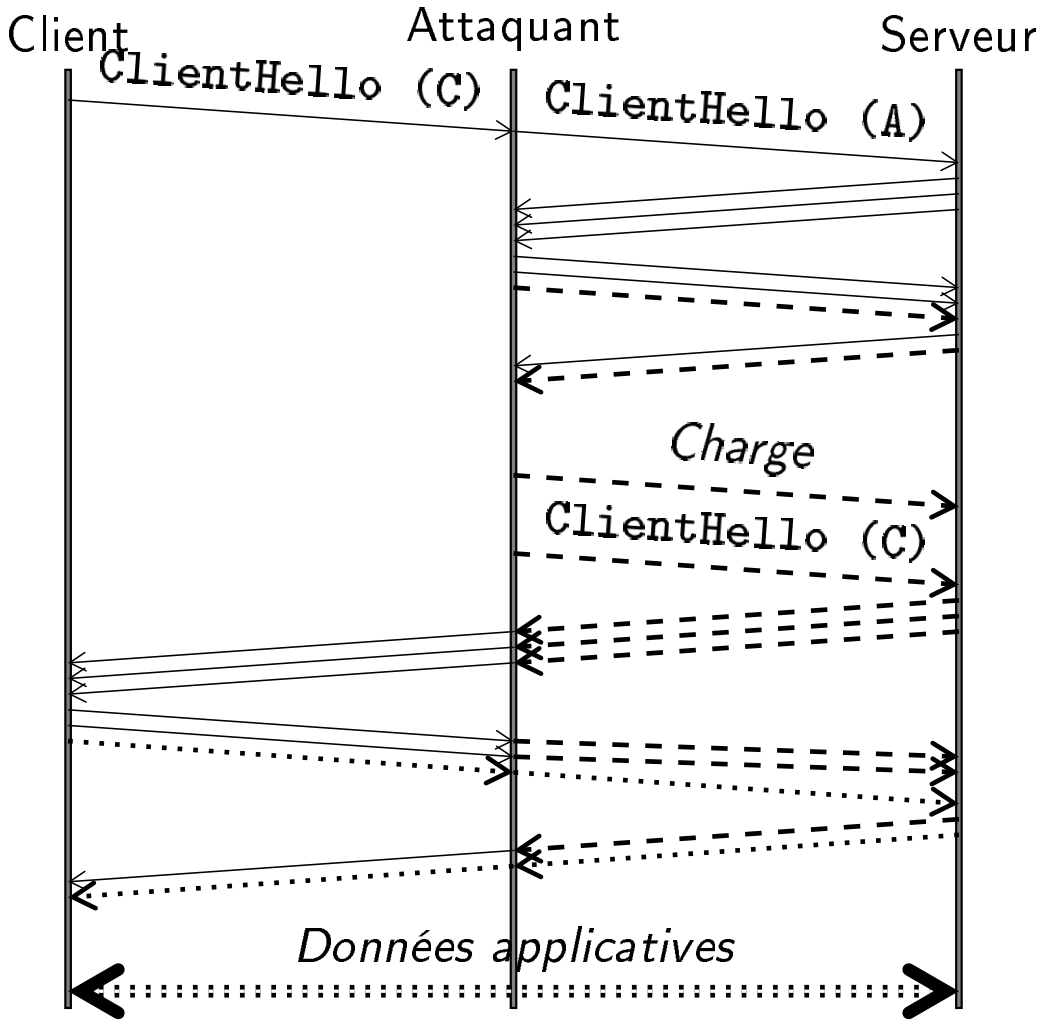
Vulnérabilité dans la renégociation



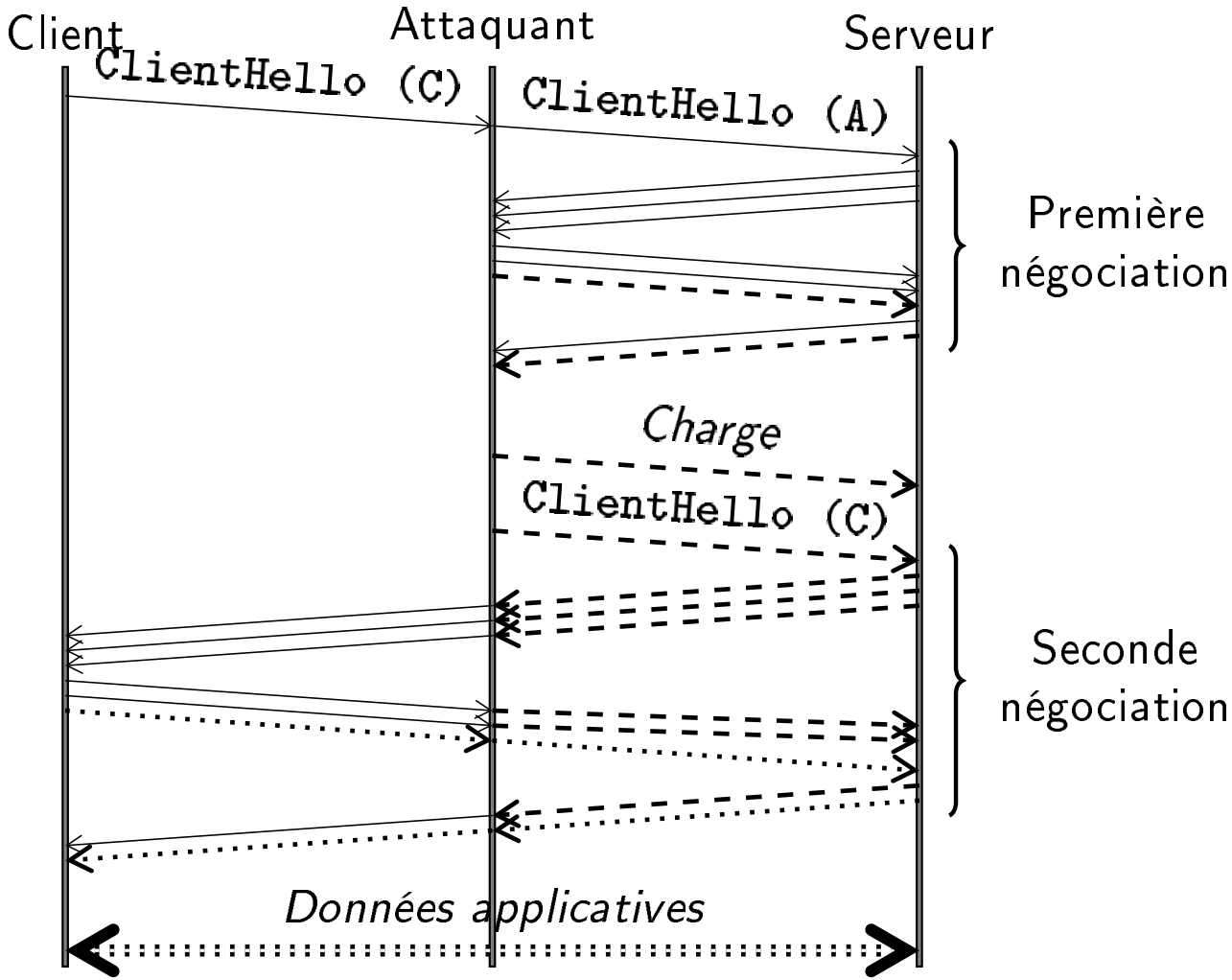
Vulnérabilité dans la renégociation



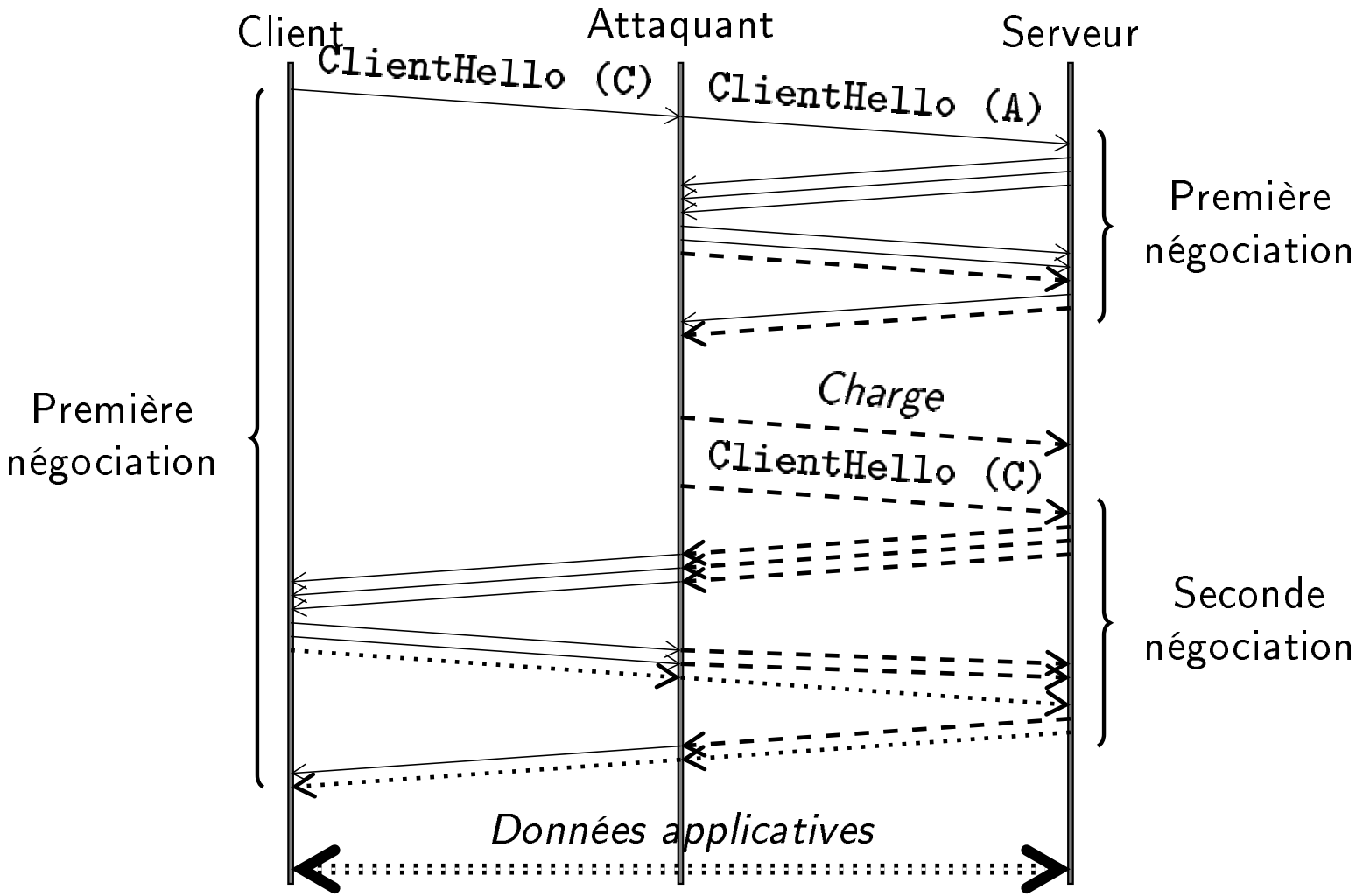
Vulnérabilité dans la renégociation



Vulnérabilité dans la renégociation



Vulnérabilité dans la renégociation



Impacts sur la sécurité d'une renégociation faible

- HTTPS est sujet à plusieurs attaques
  - Attaques proches d'une CSRF
  - Utilisation de la méthode TRACE
  - Redirection vers une page HTTP
  - Vol d'authentifiants
- SMTPS / FTPS : des pistes mais rien de concret

RFC 5746

- Correction apportée à la renégociation par la RFC 5746
- ajout d'une extension décrivant les négociations passées
- nécessite les extensions TLS

- incompatibilités notoires d'implémentations SSLv3 avec les extensions TLS
- conclusion : TLSv1.0 minimum avec RFC 5746

Implémentations

Logiciel	SSLv2	SSLv3	TLSv1.0	TLSv1.1	TLSv1.2
OpenSSL				1.0.1	1.0.1
GnuTLS					
NSS					
IE sous 7					
IE sous XP					
Firefox					
Chrome				21	
Opera					
Safari					
IIS sous 2008					
IIS sous 2003					
Apache mod_ssl				1.0.1	1.0.1

Non supporté | Configurable | Supporté

Comment tester les navigateurs et serveurs

- Démos

Mesures (1/2)

- En juillet 2011, énumération des hôtes IPv4 sur le port 443
- si l'hôte répond, on lui envoie plusieurs `ClientHello` SSL divers
  - différentes versions (SSLv2, TLSv1.0, TLSv1.2)
  - différentes suites crypto (standard, DHE, EC)
  - différentes extensions

Mesures (2/2)

Quelques résultats :

- sur les 2 milliards d'IP routables
- 26 218 653 répondent sur TCP 443
- 11 469 062 répondent avec un `ServerHello` à au moins un stimulus (soit 43,75 %)

- parmi ceux-ci,
  - 98,19 % acceptent un CH standard TLSv1.0 (005)
  - 38,55 % acceptent un CH DHE (004)
  - 39,53 % acceptent un CH SSLv2 (007)
  - 99,06 % acceptent un CH SSLv2-TLSv1.0 (008)
  - 74,58 % acceptent un CH TLSv1.2 (009)

Conclusions et mise en perspective :

- tout n'est pas perdu
- pertinence de ce type de mesures ?

Description d'une suite crypto

Une suite cryptographique décrit les algorithmes

- d'authentification (du serveur)
- d'échange de clé
- de chiffrement des données
- de protection en intégrité des données

Traditionnellement, on regroupe

- les parties `Kx` et `Au` d'une part
- les parties `Enc` et `Mac` d'autre part

Exemples

`TLS_RSA_WITH_RC4_128_MD5`

- `RSA` : chiffrement RSA (échange de clé et authentification implicite)
- `RC4_128` pour le chiffrement des données
- `HMAC-MD5` pour l'intégrité des données

Exemples

`TLS_RSA_WITH_RC4_128_MD5`

- `RSA` : chiffrement RSA (échange de clé et authentification implicite)
- `RC4_128` pour le chiffrement des données
- `HMAC-MD5` pour l'intégrité des données

`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`

- `DHE` : échange de clé Diffie-Hellman...
- `RSA` : signé par RSA
- `AES_128_CBC` pour le chiffrement des données
- `HMAC-SHA1` pour l'intégrité des données

Exemples

`TLS_RSA_WITH_RC4_128_MD5`

- `RSA` : chiffrement RSA (échange de clé et authentification implicite)
- `RC4_128` pour le chiffrement des données
- `HMAC-MD5` pour l'intégrité des données

`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`

- `DHE` : échange de clé Diffie-Hellman...
- `RSA` : signé par RSA
- `AES_128_CBC` pour le chiffrement des données
- `HMAC-SHA1` pour l'intégrité des données

DHE/ECDHE assurent la PFS (*Perfect Forward Secrecy*)

Fonctionnement de la négociation

- Le client propose une liste de suites (`ClientHello`), selon ses préférences

Fonctionnement de la négociation

- Le client propose une liste de suites (`ClientHello`), selon ses préférences
- Le serveur choisit parmi cette liste

Fonctionnement de la négociation

- Le client propose une liste de suites (`ClientHello`), selon ses préférences

- Le serveur choisit parmi cette liste
- Deux cas classiques :
  - comportement courtois (Apache)
  - comportement directif (IIS)

Justification du tri

Recherche de suites offrant

- un bon niveau de sécurité
- une présence dans les implémentations

Echanges de clé

- *KRB5* : Kerberos
- *DH* et *ECDH* : Diffie-Hellman fixe (dans le certificat)
- *PSK* et *SRP* : utilisation d'un secret pré-partagé

- RSA : chiffrement RSA
- DHE et ECDHE : Diffie-Hellman éphémère

Authentification

- *NULL* : à proscrire
- *KRB5* : Kerberos
- *PSK* et *SRP* : secret pré-partagé
- *DSS* : DSA

- RSA : signature ou chiffrement RSA
- ECDSA : DSA sur courbes elliptiques

Chiffrement des données (1/2)

- *NULL* : possible, mais rarement souhaité
- *DES*, *RC2*
- *ARIA* et *SEED*

- algorithmes acceptables pour une quantité de trafic limité
  - RC4 : seul algorithme de chiffrement par flot
  - IDEA et 3DES

- AES
- CAMELLIA

Chiffrement des données (1/2)

Deux modes de chiffrement par bloc :

- CBC
- GCM (avec TLSv1.2)

Intégrité des données

- *HMAC MD5* : peu recommandable

- HMAC SHA1 : à défaut
- HMAC SHA256 ou HMAC SHA384 : disponible avec TLSv1.2
- GCM (mode *AEAD*) : disponible avec TLSv1.2

Résultat

- 39 des suites retenues sont dans OpenSSL 1.0.1
- dont 19 compatibles avec TLSv1.0

Préférences supplémentaires :

- utiliser les suites assurant la PFS
- éviter RC4 et 3DES
- éviter SHA1
- on obtient 12 suites compatibles avec OpenSSL 1.0.1

Avertissement

La taille des paramètres asymétriques n'est *pas* négociée :

- taille des clés RSA/DSA/ECDSA dans les certificats
- taille des groupes DH/ECDH lors de l'échange Diffie-Hellman

Réalisme des recommandations

Un mot sur les mesures :

- certains serveurs répondent avec une suite non proposée
- d'autres renvoient un serveur invalide car les deux octets manquent.
- conclusion : il existe des piles TLS exotiques dans la nature

Réalité des connexions

- Statistiques sur les données de l'EFF :
  - 38 % TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  - 32 % TLS_RSA_WITH_RC4_128_MD5
  - 23 % TLS_RSA_WITH_AES_256_CBC_SHA

Réalité des connexions

- Statistiques sur les données de l'EFF :
  - 38 % `TLS_DHE_RSA_WITH_AES_256_CBC_SHA`
  - 32 % `TLS_RSA_WITH_RC4_128_MD5`
  - 23 % `TLS_RSA_WITH_AES_256_CBC_SHA`
- Conséquence du comportement directif de certains serveurs
  - IIS choisira toujours `TLS_RSA_WITH_RC4_128_MD5`
  - le comportement est configurable, mais *global* au système sous Windows

Réalité des connexions

- Statistiques sur les données de l'EFF :
  - 38 % `TLS_DHE_RSA_WITH_AES_256_CBC_SHA`
  - 32 % `TLS_RSA_WITH_RC4_128_MD5`
  - 23 % `TLS_RSA_WITH_AES_256_CBC_SHA`
- Conséquence du comportement directif de certains serveurs
  - IIS choisira toujours `TLS_RSA_WITH_RC4_128_MD5`
  - le comportement est configurable, mais *global* au système sous Windows
- Est-il possible d'imposer la PFS du point de vue client ?
  - ne proposer que des suites DHE/ECDHE fait perdre des serveurs
  - cette configuration globale est à l'application, voire au système
  - besoin d'affiner la décision par application et par serveur

Message Certificate

- Le message contient une suite de certificats
- le premier désigne le serveur TLS
- chaque certificat est signé par le suivant
- l'autorité racine peut être omise

- la désignation du serveur dans le certificat est mal définie
  - HTTPS : `Subject.CommonName` historiquement
  - HTTPS : Extension `SubjectAltName` conseillée
  - SMTPS : Problème en cas de serveurs multiples : que doit contenir le certificat
  - chaque cas est traité différemment

La réalité des chaînes de certification

- Chaînes incomplètes
- chaînes désordonnées
- certificats dupliqués
- certificats inutiles
- chaînes multiples (A, B, C, $B^\star$, $C^\star$)

- incompatibilités réelles entre certaines piles et certains sites

La réalité des autorités de certifications

Ce n'est pas une forêt !

- signatures croisées entre autorités
- besoin de garder en cache certains certificats d'autorité (pour compléter les chaînes)

La réalité des autorités de certifications

Ce n'est pas une forêt !

- signatures croisées entre autorités
- besoin de garder en cache certains certificats d'autorité (pour compléter les chaînes)

De nombreux magasins... bien remplis

- Firefox en contient au moins trois sortes :
  - les certificats racines par défaut (environ 150)
  - les certificats du profil
    - ajoutés par l'utilisateur
    - ajoutés (sans notion de confiance) par Firefox
  - un cache de certificats de session pour stocker d'autres certificats rencontrés
- Microsoft propose un magasin dynamique
- Idem pour Opera

Suite sur les magasins

- Beaucoup d'applications utilisent un magasin indépendant
  - produits Mozilla
  - Safari
  - Opera
  - applications Adobe
  - Java (parfois)

Suite sur les magasins

- Beaucoup d'applications utilisent un magasin indépendant
  - produits Mozilla
  - Safari
  - Opera
  - applications Adobe
  - Java (parfois)
- D'autres utilisent des magasins partagés
  - applications Microsoft
  - Chrome
  - produits Mozilla, à l'avenir (`$HOME/.pki`)

Suite sur les magasins

- Beaucoup d'applications utilisent un magasin indépendant
  - produits Mozilla
  - Safari
  - Opera
  - applications Adobe
  - Java (parfois)
- D'autres utilisent des magasins partagés
  - applications Microsoft
  - Chrome
  - produits Mozilla, à l'avenir (`$HOME/.pki`)

- Pour ou contre un magasin centralisé
  - avantages : retrait efficace d'autorité compromise, ajout rapide d'autorités internes
  - inconvénient : manque de finesse dans la politique de confiance

Causes

- Taille de clé faible
  - TI
  - AC malaise Digicert Sdn. Bhd.
- Mauvaise gestion de l'accès aux secrets
  - accès physique au serveur
  - accès logique (vulnérabilité du serveur, mauvais contrôle d'accès)
- Défaut dans la génération d'aléa
  - 2006-08 : Debian OpenSSL
  - 2011 : Lenstra et al. — *Ron was wrong, Whit is right*

Impacts

La perte d'une clé privée de serveur permet

- une attaque passive sur les connexions utilisant le chiffrement RSA, y compris les communications passées (pas de PFS)
- une attaque active dans tous les cas

La perte d'une clé privée d'une AC reconnue par les navigateurs permet

- la signature de certificats pour des sites quelconques (y compris `mail.google.com`)
- la signature de certificats d'autorité
- la signature de code (pilotes de périphériques, applets Java...)
- Remarque : en général, la clé privée n'est pas divulguée, mais un attaquant gagne le droit de l'utiliser, ce qui restreint ses possibilités.

- mars 2011 : Comodo
  - intrusion d'un attaquant
  - signature de certificats pour des sites intéressants
  - détection, révocation et mise à jour des navigateurs dans les jours suivants

- mars 2011 : Comodo
  - intrusion d'un attaquant
  - signature de certificats pour des sites intéressants
  - détection, révocation et mise à jour des navigateurs dans les jours suivants
- juillet-septembre 2011 : Diginotar
  - intrusion d'un attaquant
  - détection sans correction de la faille sous-jacente
  - signature de nombreux certificats
  - détection en Iran de l'utilisation de certificats frauduleux
  - prise de conscience chez Diginotar et le gouvernement NL
  - *révocation* de l'autorité racine
  - faillite de Diginotar

- mars 2011 : Comodo
  - intrusion d'un attaquant
  - signature de certificats pour des sites intéressants
  - détection, révocation et mise à jour des navigateurs dans les jours suivants
- juillet-septembre 2011 : Diginotar
  - intrusion d'un attaquant
  - détection sans correction de la faille sous-jacente
  - signature de nombreux certificats
  - détection en Iran de l'utilisation de certificats frauduleux
  - prise de conscience chez Diginotar et le gouvernement NL
  - *révocation* de l'autorité racine
  - faillite de Diginotar
- janvier 2012 : Trustwave
  - annonce de la certification d'AC intermédiaire pour *analyser de manière transparente le trafic*
  - certificats révoqués
  - nombreuses réactions dans la communauté, mais l'AC reste dans les magasins

Solutions actuelles

- Restreindre les autorités racines
  - facile côté serveur
  - faisable pour les clients VPN, mail, etc.
  - pas envisageable pour les navigateurs

- Audit des autorités de certification
- utilisation de HSM pour protéger les clés privées
- ajout de contraintes sur les certificats générés
  - interdire la signature automatique d'AC intermédiaire
  - interdire l'usage de certains noms de domaines
  - utiliser des extensions X.509 pour restreindre la portée des ACs

- les mécanismes officiels de révocation (CRL, OCSP)
- des listes noires
- des listes blanches (*Certificate pinning* dans Chrome)

Réflexions en cours

- améliorer l'existant
  - certificats EV (réalité surtout commerciale)
  - CA/B Forum
  - utilisation d'extensions X.509

Réflexions en cours

- améliorer l'existant
  - certificats EV (réalité surtout commerciale)
  - CA/B Forum
  - utilisation d'extensions X.509
- forcer la révocation (*hard fail*)
  - OCSP *stapling*
  - certificats à durée de vie courte

Réflexions en cours

- améliorer l'existant
  - certificats EV (réalité surtout commerciale)
  - CA/B Forum
  - utilisation d'extensions X.509
- forcer la révocation (*hard fail*)
  - OCSP *stapling*
  - certificats à durée de vie courte
- ajouter de l'information supplémentaire hors-bande
  - DANE (*DNS-Based Authentication of Named Entities*) poussé par Google
  - Convergence de Moxie Marlinspike
  - Sovereign Keys de l'EFF
  - TACK

Réflexions en cours

- améliorer l'existant
  - certificats EV (réalité surtout commerciale)
  - CA/B Forum
  - utilisation d'extensions X.509
- forcer la révocation (*hard fail*)
  - OCSP *stapling*
  - certificats à durée de vie courte
- ajouter de l'information supplémentaire hors-bande
  - DANE (*DNS-Based Authentication of Named Entities*) poussé par Google
  - Convergence de Moxie Marlinspike
  - Sovereign Keys de l'EFF
  - TACK
- permettre plus de finesse dans la gestion des magasins
  - par application
  - par serveur

Conclusion

- SSL/TLS est protocole mûr basé sur des schémas éprouvés et il est théoriquement possible de l'employer correctement

Conclusion

- SSL/TLS est protocole mûr basé sur des schémas éprouvés et il est théoriquement possible de l'employer correctement

- Si on maîtrise clients et serveurs, mise en pratique sécurisée possible
- Pour les navigateurs, beaucoup de difficultés
  - équilibre difficile entre compatibilité et sécurité
  - mécanisme de gestion de la confiance douteux

Conclusion

- SSL/TLS est protocole mûr basé sur des schémas éprouvés et il est théoriquement possible de l'employer correctement

- Si on maîtrise clients et serveurs, mise en pratique sécurisée possible
- Pour les navigateurs, beaucoup de difficultés
  - équilibre difficile entre compatibilité et sécurité
  - mécanisme de gestion de la confiance douteux

- Perspectives :
  - proposer des plates-formes de test (client/serveur)
  - continuer l'analyse du paysage SSL
  - travail sur la confiance et la révocation (CA/B Forum, OCSP stapling, DANE, TACK...)
  - IDS/IPS : utilisation de Suricata
  - proxy local pour affiner la gestion de la confiance

Questions ?

Merci de votre attention.