Soutenance projet annuel - Audit des implantations SSL/TLS

Claire Smets – William Boisseleau – Pascal Edouard – Mathieu Latimier – Julien Legras

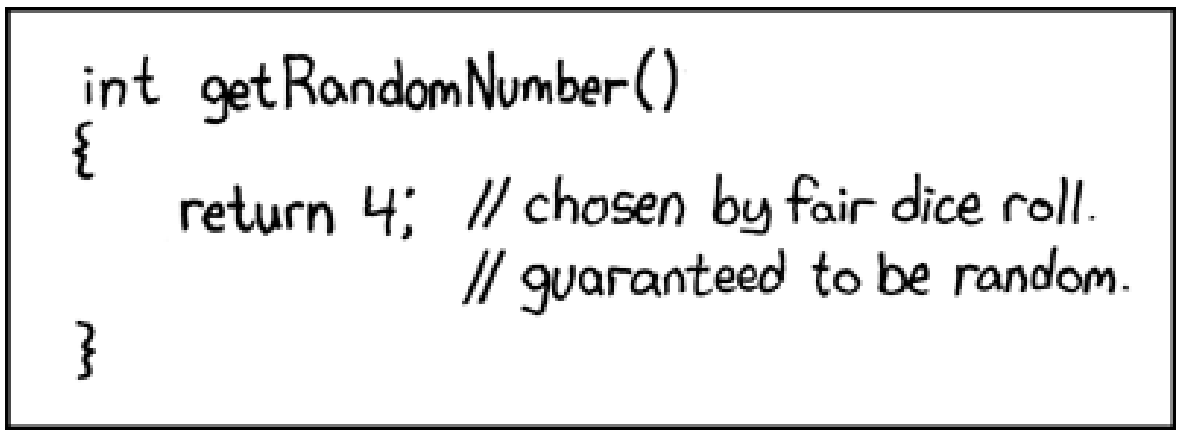
Master 2 Sécurité des Systèmes Informatiques

28/02/2014

Sujet et problématique I

Motivations

- incertitudes cryptographiques liées aux récents scandales ;
- 2012 – étude de l'Université du Michigan sur la sécurité d'internet : *Mining your Ps and Qs : Widespread Weak Keys in Network Devices*



Sujet et problématique II

Projet

- mesure de l'évolution par rapport à cette étude ;
- identification des problèmes avérés ;
- audit de la principale bibliothèque : OpenSSL.

Exigences du client

- audit des certificats RSA ;
- données importantes pour résultats représentatifs (500 000 certificats).

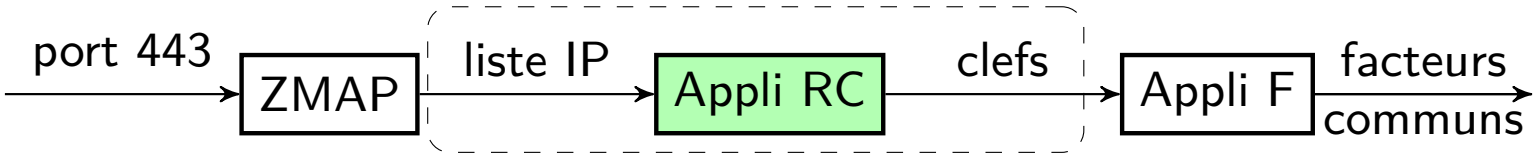
Sommaire

Récupération des adresses

ZMAP

- open source dévoloppé par l'équipe de l'Université du Michigan ;
- scanneur de ports : jusqu'à 1,4 millions de paquets SYN par seconde.

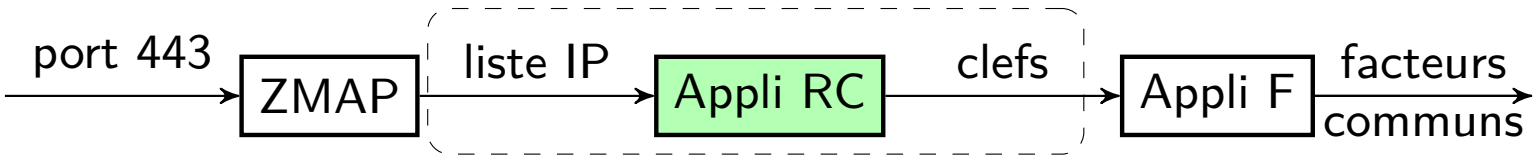
Récupération des certificats I



Récupération des certificats

- commandes openssl pour établir la connexion SSL ;
- enregistrement de la session ;
- extraction des certificats et des clefs RSA de la session SSL ;
- élimination des doublons.

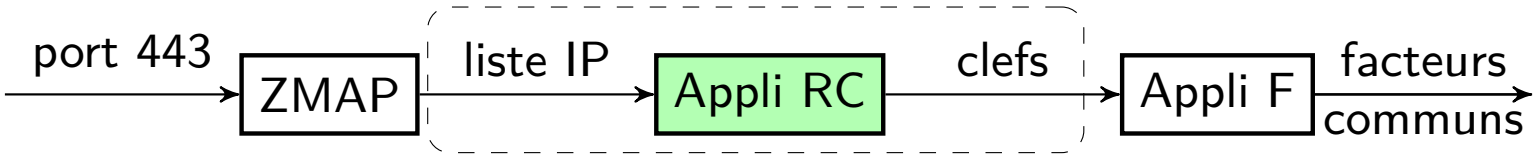
Certificats récupérés

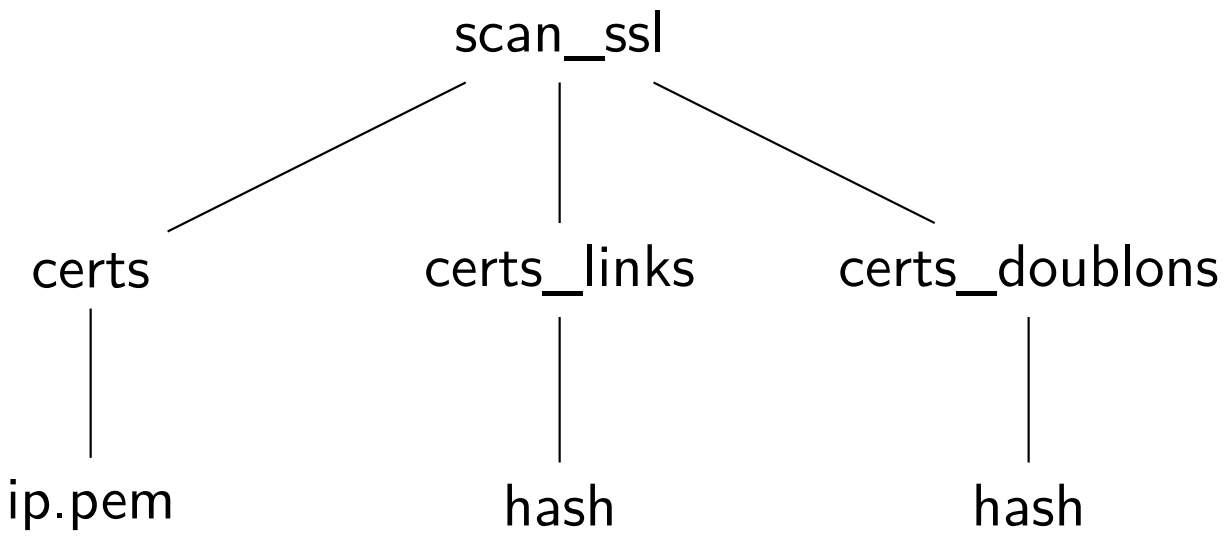


Certificats mal formés

- *Common Name* vide : validation impossible au regard du nom de domaine ;
- *Serial Number* seul : absence d'informations sur le certificat.

Gestion des doublons





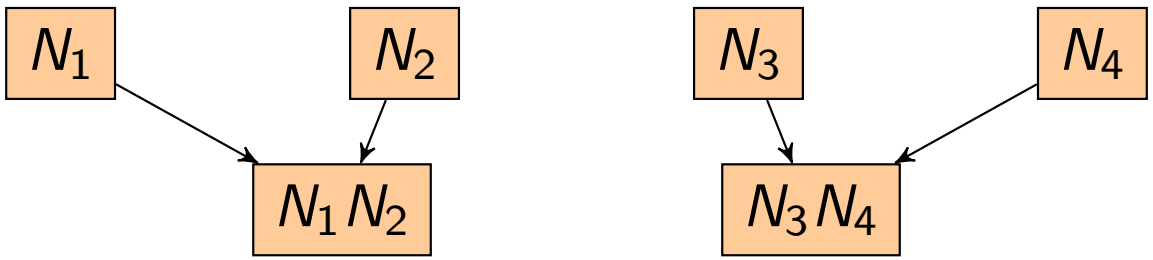
Factorisation

PGCD deux à deux

N_1 N_2 N_3 N_4

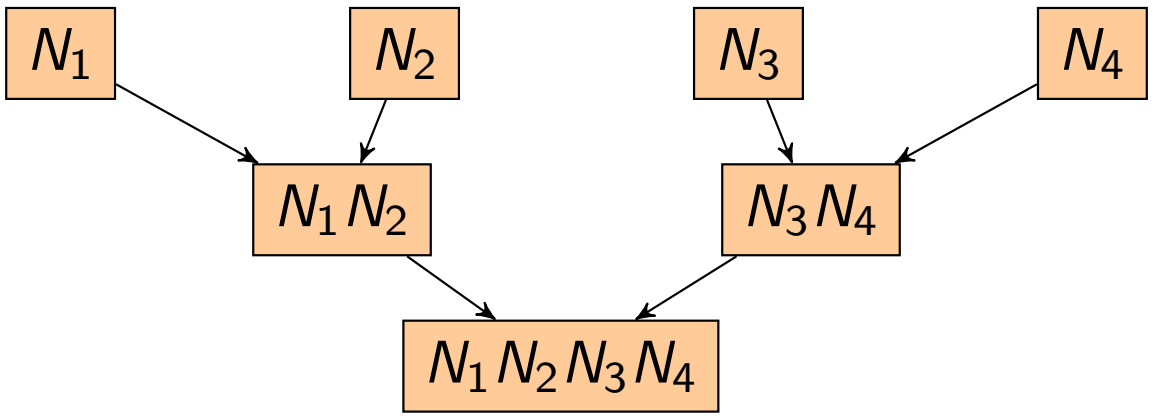
Factorisation

PGCD deux à deux



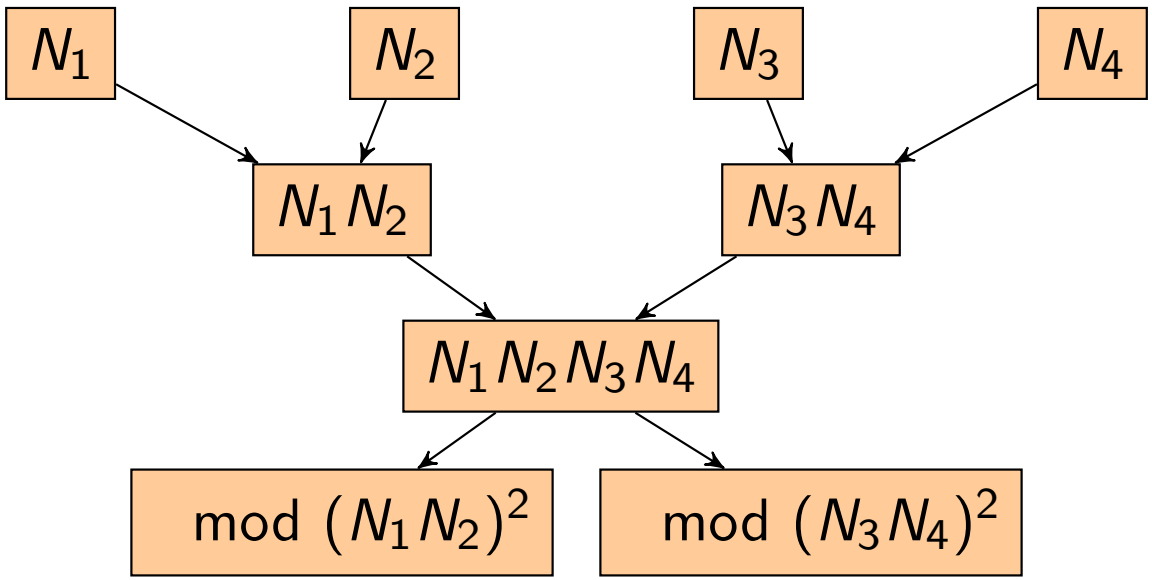
Factorisation

PGCD deux à deux



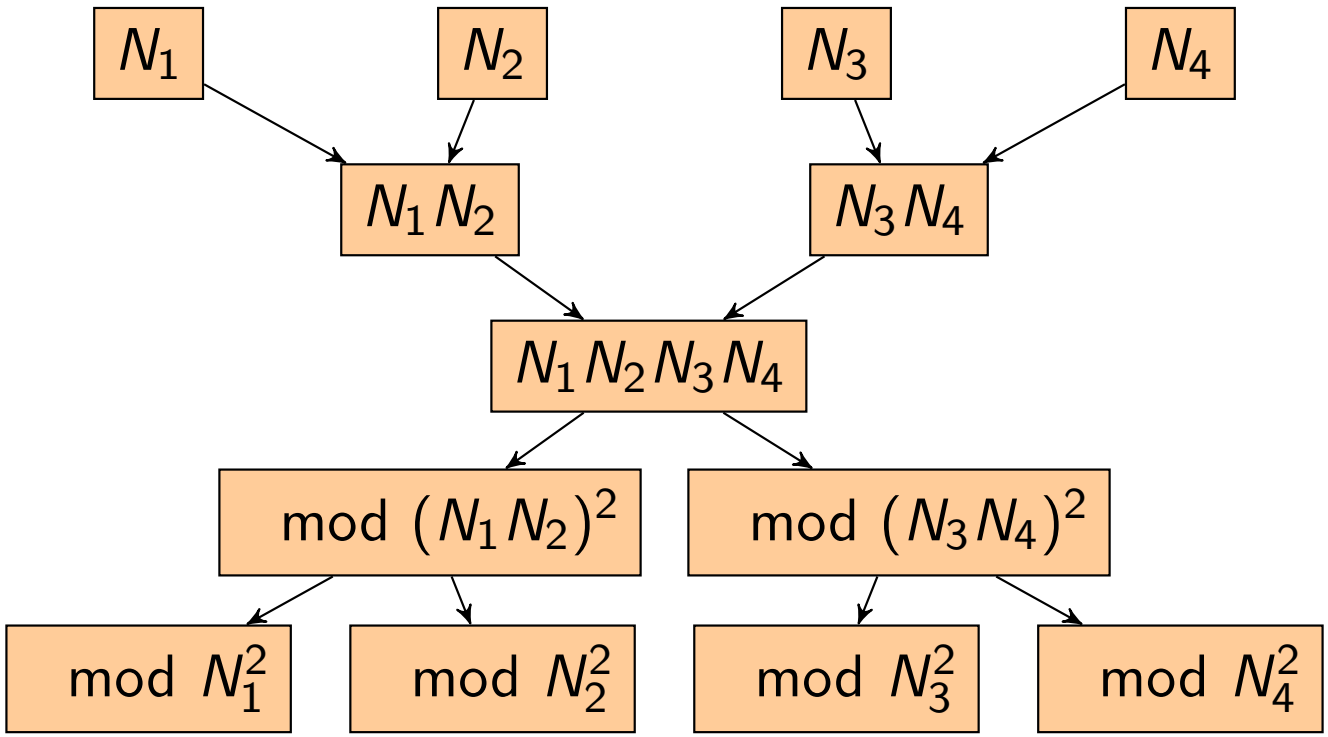
Factorisation

PGCD deux à deux



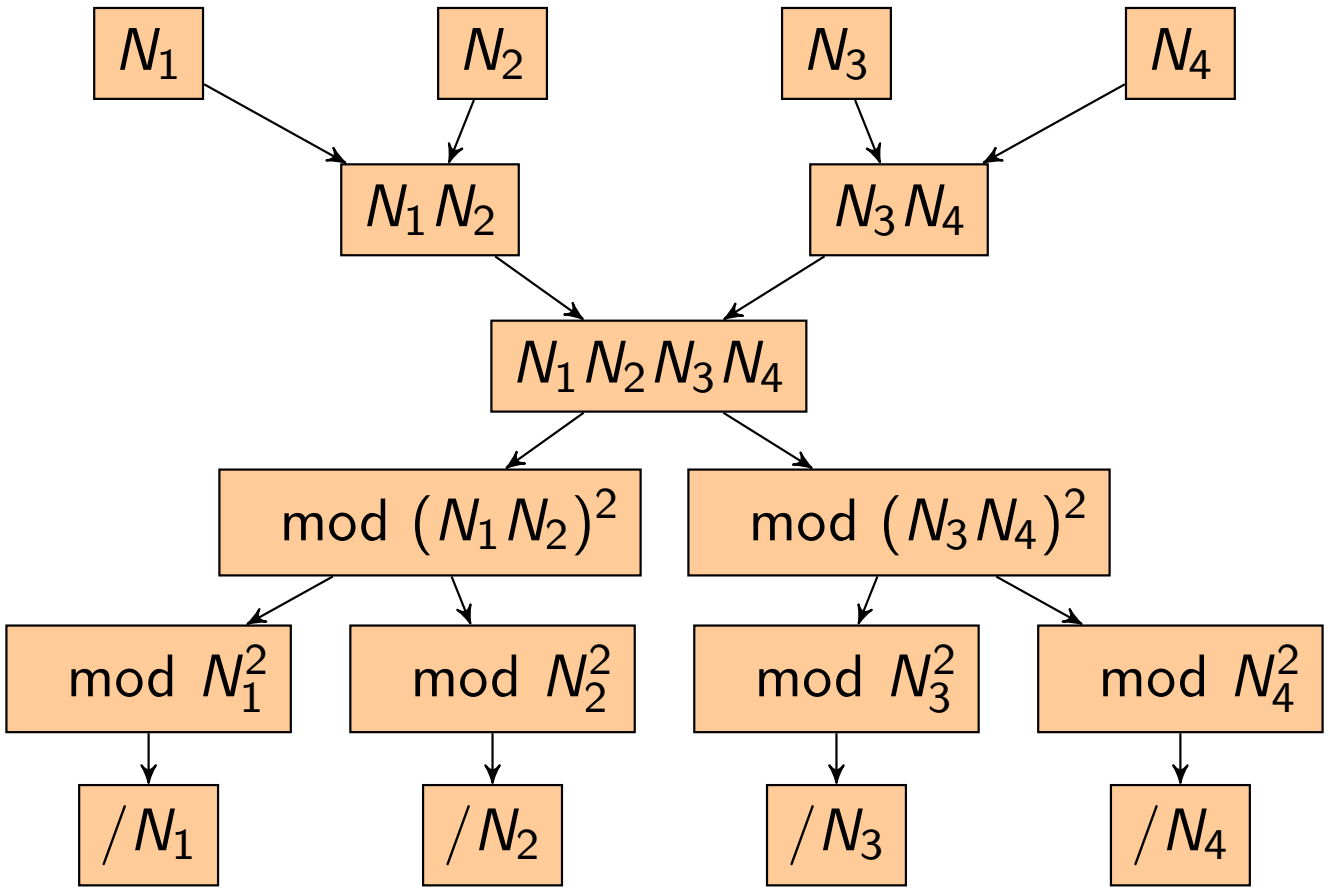
Factorisation

PGCD deux à deux



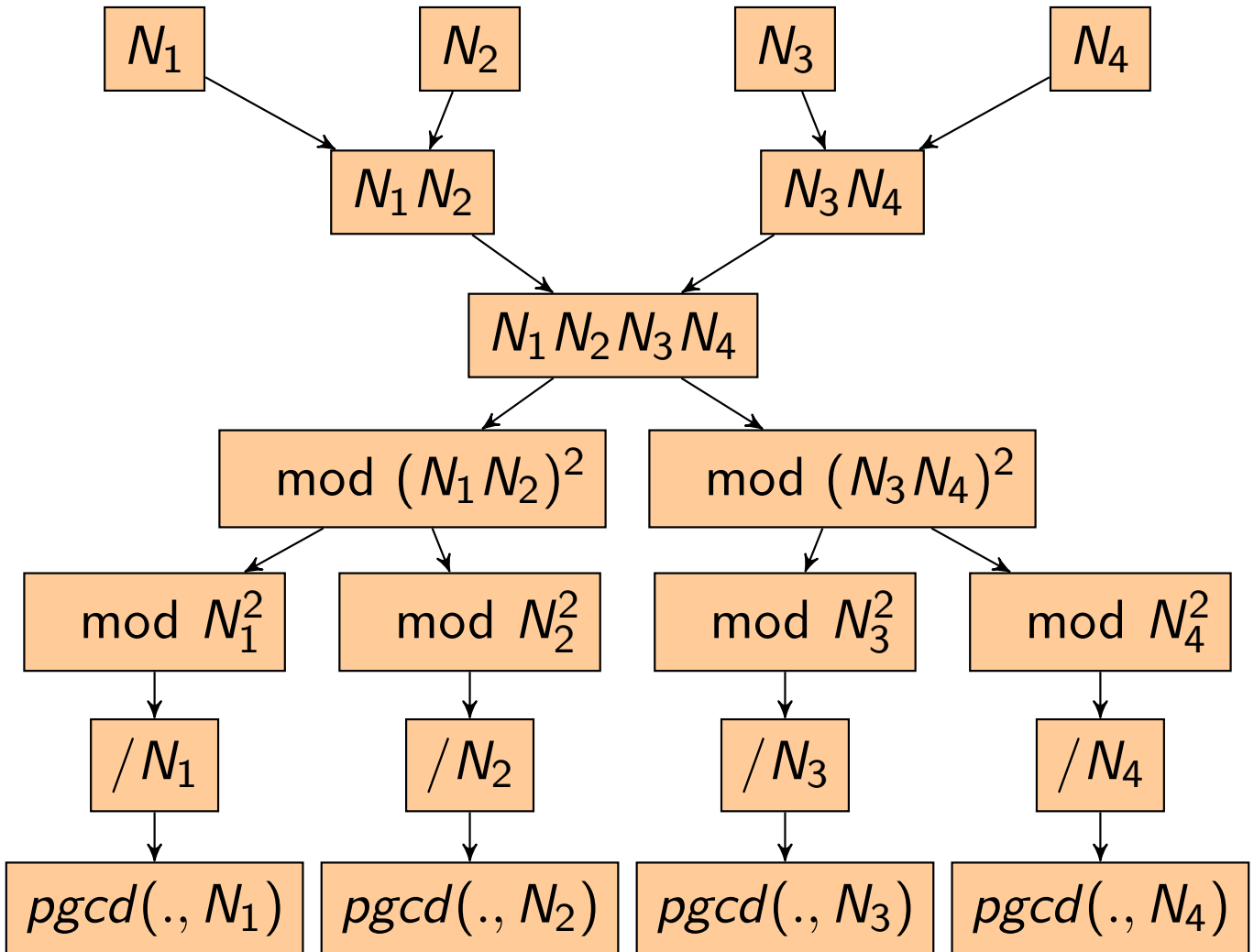
Factorisation

PGCD deux à deux



Factorisation

PGCD deux à deux



Factorisation

Exemple

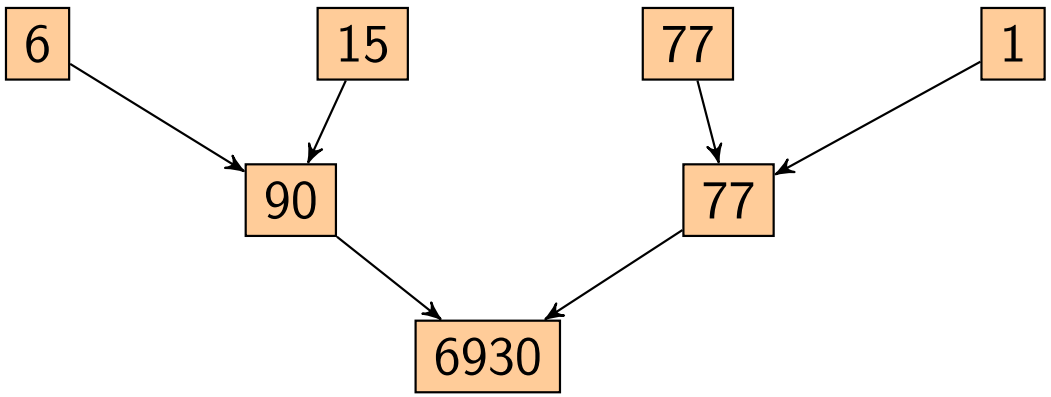
6 15 77 1

Factorisation

Exemple

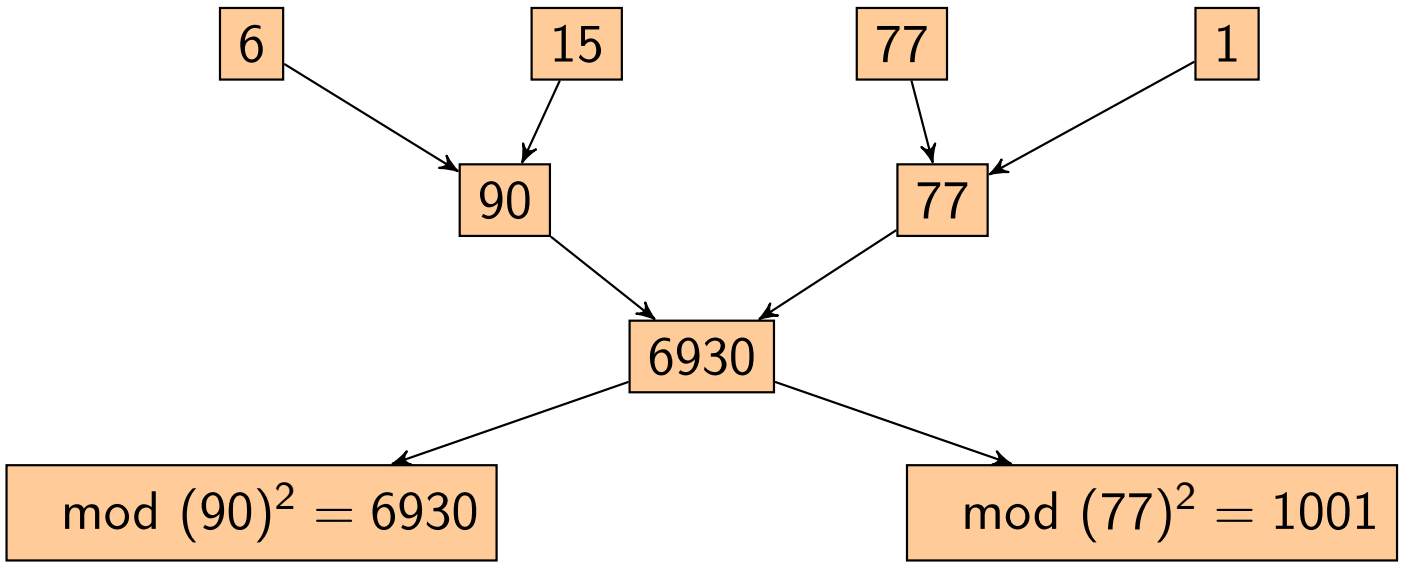
Factorisation

Exemple



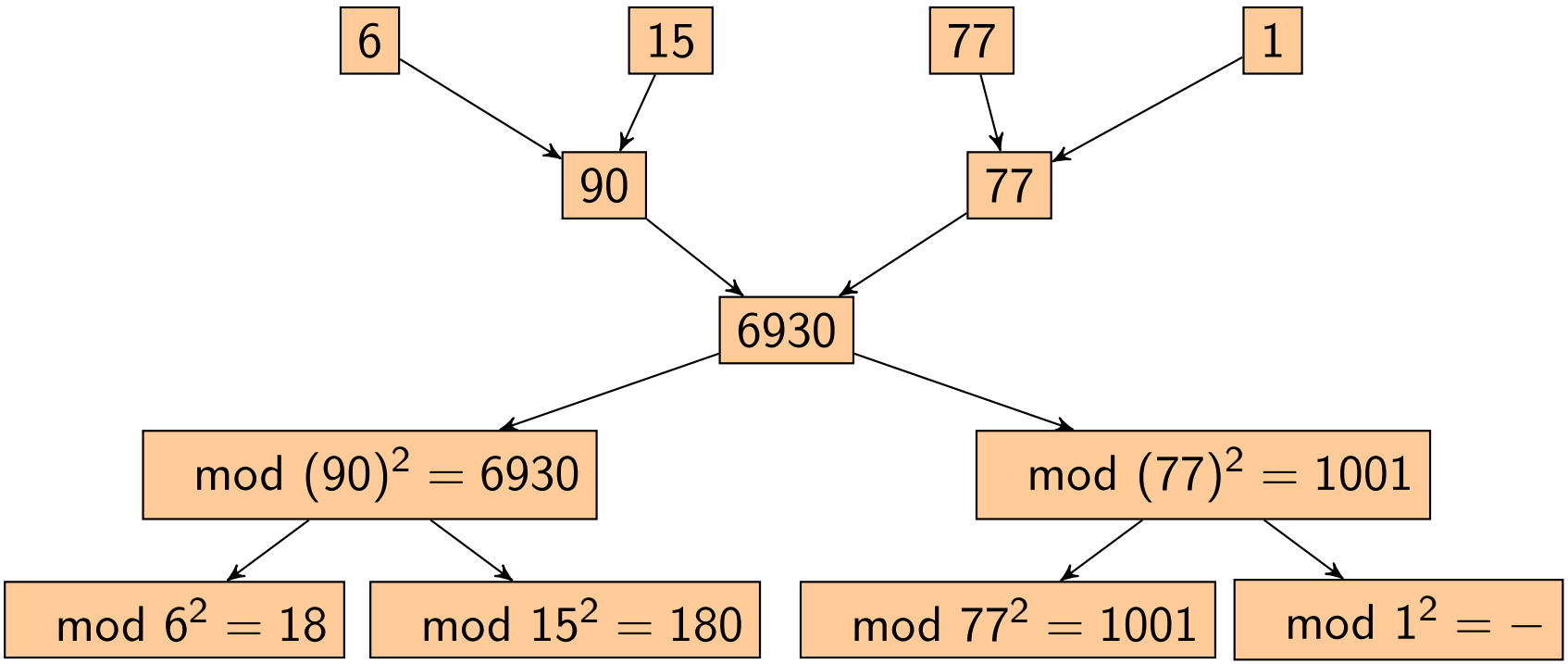
Factorisation

Exemple



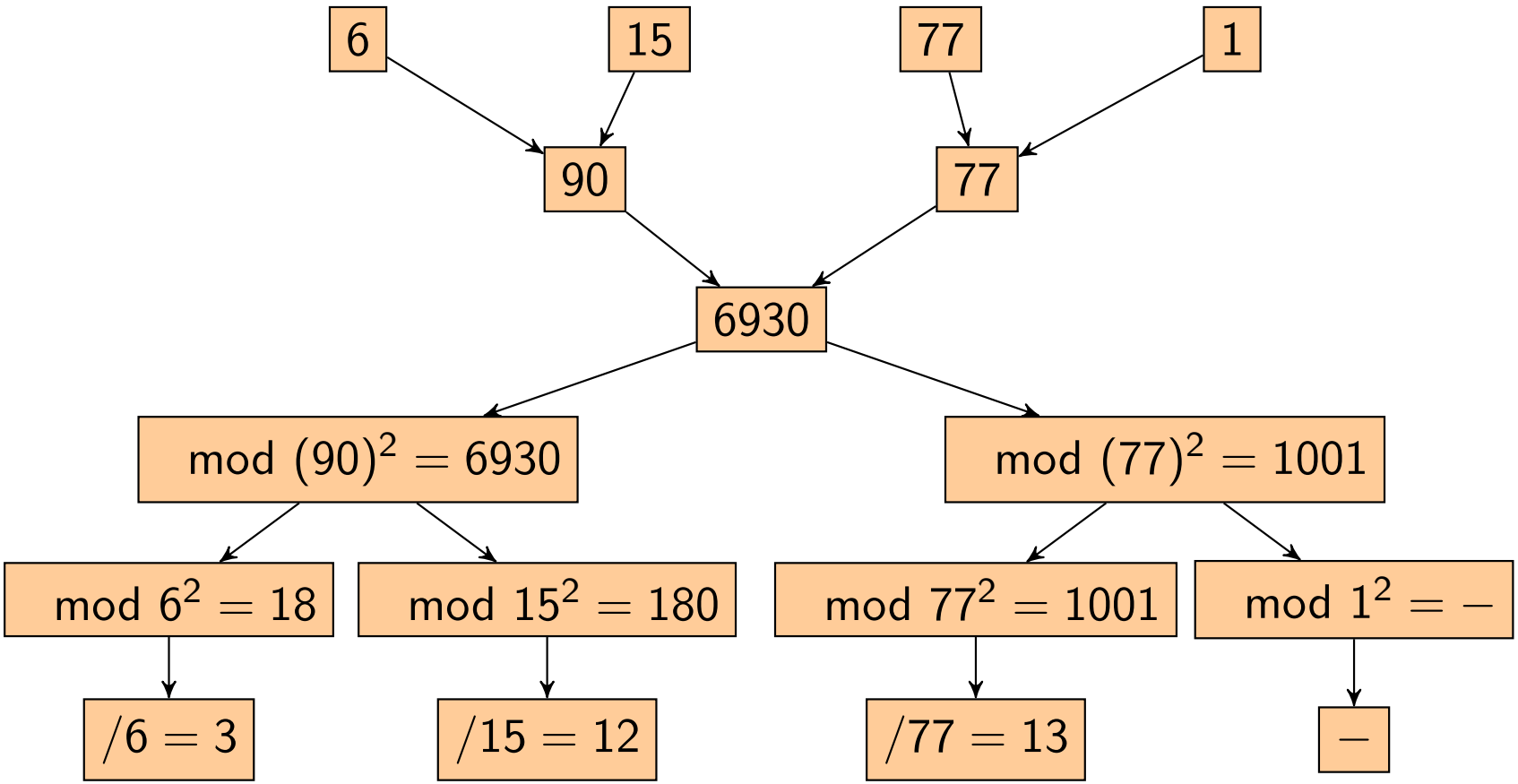
Factorisation

Exemple



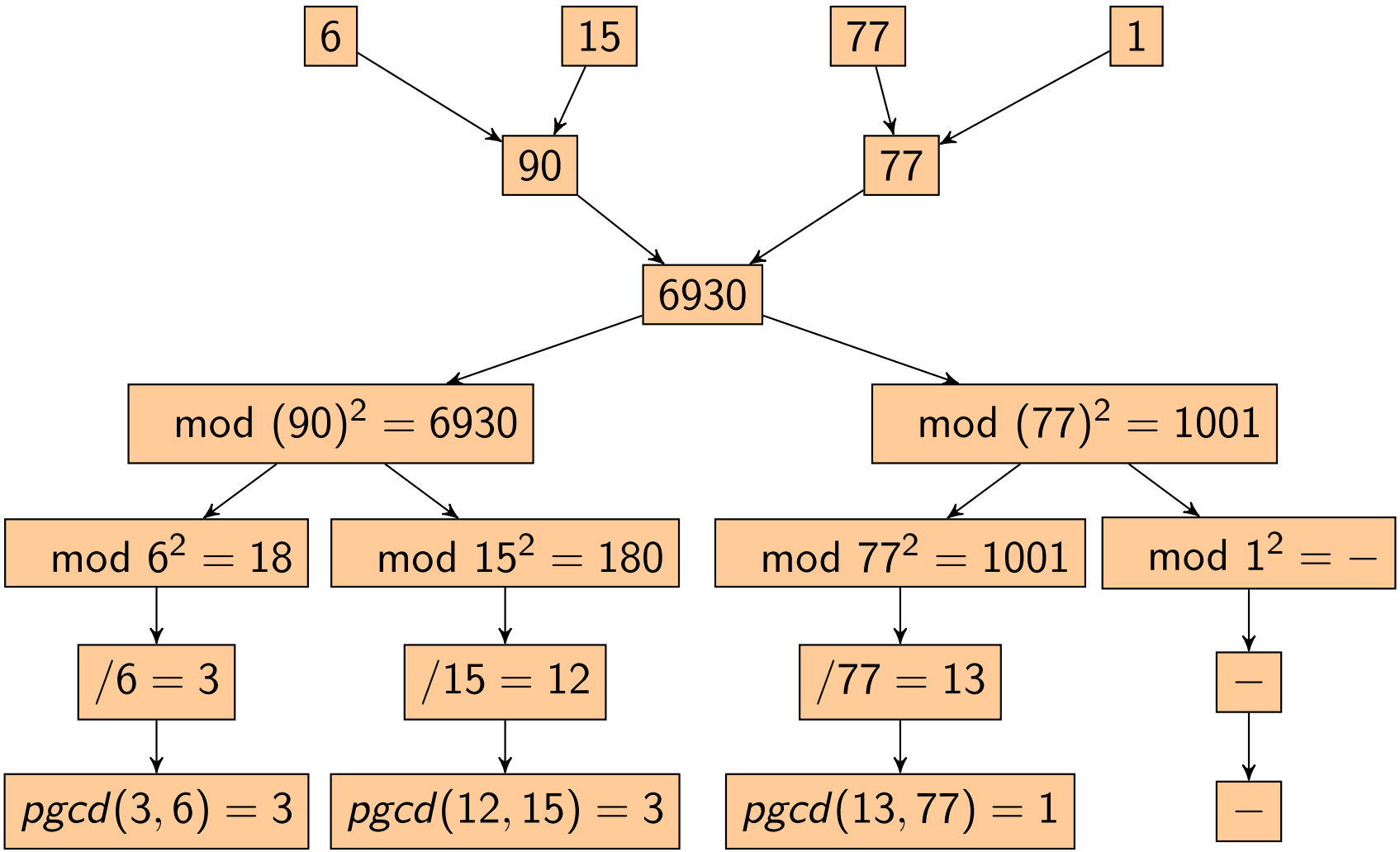
Factorisation

Exemple



Factorisation

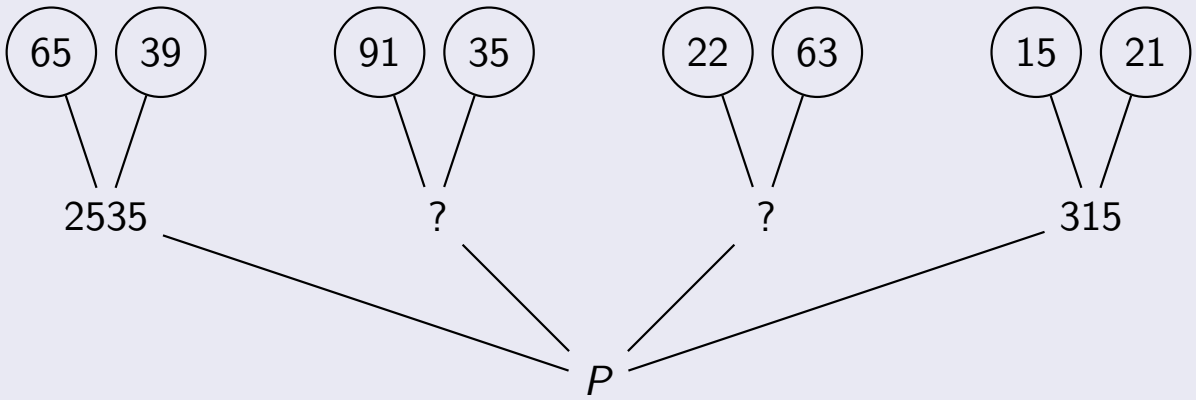
Exemple



Optimisations I

Parallélisation

En largeur avec 100 threads



Parallélisation

En hauteur : ralentissement du traitement à cause de la dépendances entre les niveaux

Optimisations II

Langage et bibliothèque

C & *libGMP*

Stockage

- programme initial : utilisation de fichiers pour stocker chaque niveau ;
- programme amélioré : stockage de l'arbre des produits en RAM (4-5 Go pour 500 000 clefs).

  $\Rightarrow$ jusqu'à 10 fois plus rapide même avec un SSD

Calcul des carrés

Préférer la fonction d'exponentiation de GMP plutôt que la multiplication $\Rightarrow$ jusqu'à 5 fois plus rapide

Factorisation

Démonstration

Jeu de données :

`15, 22, 437, 1189, 527, 21, 3233, 3827`

Résultats

Base de données

- récupération des facteurs communs - clefs publiques ;
- scripts perl & requêtes SQL.

Statistiques

- émetteurs ;
- 99.6% (R) - 0.4% (V) ;
- critères de recherche :
  - taille de clefs ;
  - émetteurs, sujets : CISCO - 6280 (R) - 40 (V).

Introduction

Contexte

- origine de vulnérabilité des certificats ;
- contexte actuel : sentiment d'incertitude ;
- beaucoup d'outils utilisés.

Audit d'OpenSSL

Cinq grands axes :

- la génération de l'aléatoire ;
- la génération des clefs ;
- le chiffrement et les protocoles ;
- les signatures et les authentifications ;
- les protocoles SSL et TLS.

Failles

Où ?

- site des CVE ;
- site de Vigil@nce ;
- RFC et standards.

Quand ?

- ces dernières années ;
- failles plus anciennes.

Corrections ?

Présence de corrections ? Si oui, lesquelles et par qui ?

OpenSSL

Observation

Développé par Eric Young & Tim Hudson :

- documentation ;
- code développé en mode réactif ;
- lisibilité du code ;
  - exemple de code

    `n=((p[0]&0x7f)<<8)|p[1];`
  - nom des fichiers : `s3_` ;
- commentaires des commits.

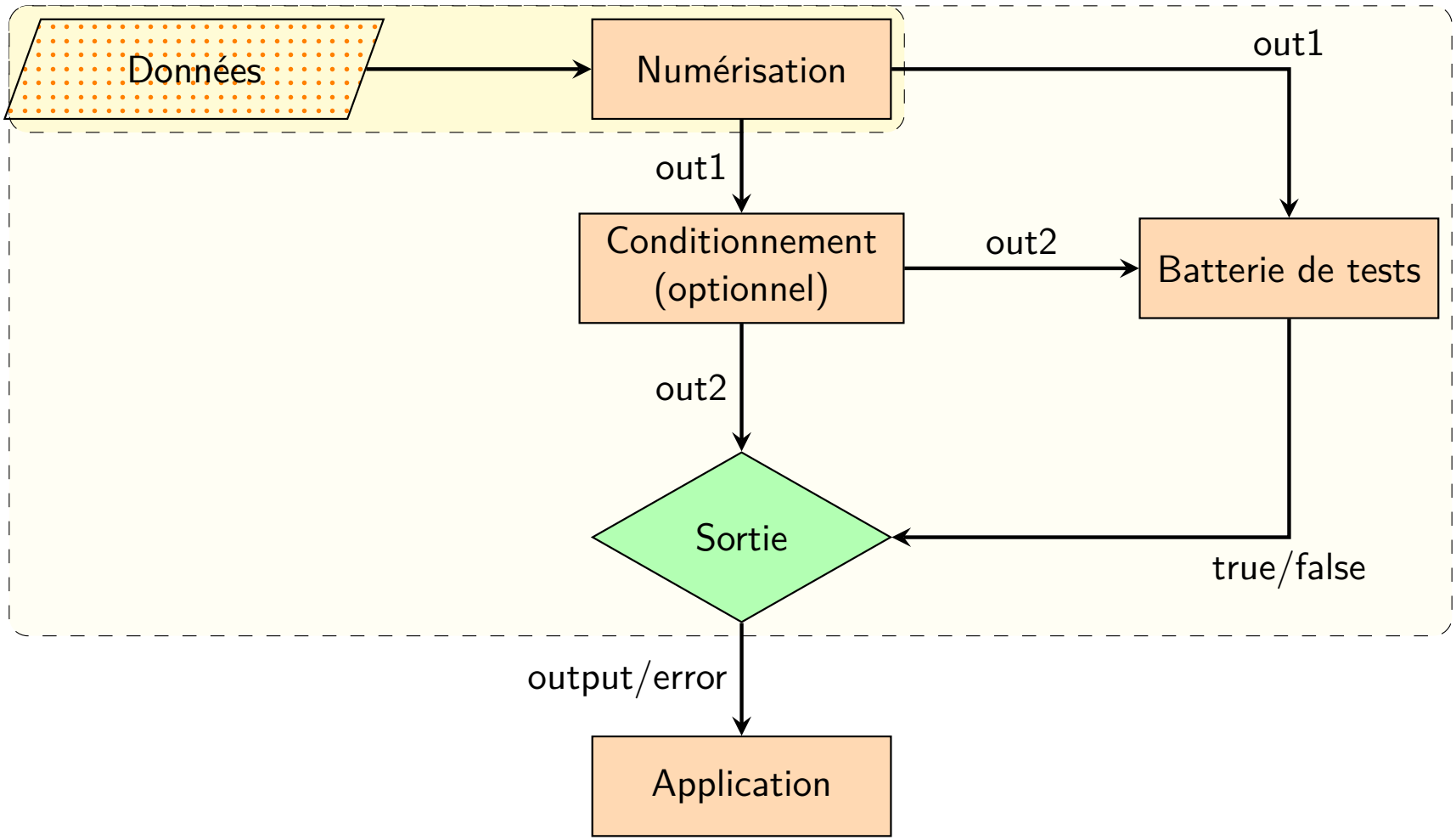
Générateur aléatoire

Définitions

Générateur

- problème aléatoire ;
- mesure de l'entropie :
  $H(X) = -\sum_x P(X = x) * \log_2(P(X = x))$;
- entropie et moduli.

Entropie


Données
Numérisation
out1
out1
Conditionnement
(optionnel)
out2
Batterie de tests
out2
Sortie
true/false
output/error
Application

Entropie

Tests

- tests trop anciens, `FIPS 140-1` (1994), non mis à jour :
  - test monobit ;
  - test poker ;
  - test runs ;
  - test long runs.
- autres tests proposés dans le rapport.

Entropie – Démonstration

Faille Debian 4.0 sous OpenSSL 0.9.8

Après avoir récupéré l'ensemble des certificats sur l'Internet, on peut identifier rapidement ceux qui ont étés générés durant la faille Debian/OpenSSL entre 2006 et 2008.

- **durée de l'attaque** : quelques heures ;
- **conséquences** : forger de faux certificats, de fausses signatures et déchiffrer des messages privées ;
- **qui ?** : grandes entreprises (e.g. IBM, CISCO), routeurs, universités, etc. ;
- **fin de validité de certificats** : 2020 – 2030.

Génération des clefs

Principes de Kerchkoff

- Le *secret* réside dans la clef ;
- les algorithmes de génération de clefs ne doivent donner aucune information sur la clef.

Deux grands types de générateur de clés

- Générateurs de bits aléatoires (RGB) pour les clés privées de certains algorithmes (i.e. AES, DSA) ou pour le salage (i.e. *seed* de RSA)
- Générateurs de clefs asymétriques, qui utilisent des fonctions à sens uniques, largement diffusées et ne devant délivrer aucune information sur le secret.

Génération des clefs II

Audit : Diffie-Hellman Ephémère en mode FIPS

- **Description** : Un attaquant écoutant une communication chiffré en SSL/TLS entre un client et un serveur peut déchiffrer tout les messages en forçant la génération d'un secret Diffie-Hellman prédictible.
- **Comment ?** : En modifiant le trafic réseau par exemple.
- **Pourquoi ?** : L'activation du mode FIPS ne rejette pas les paramètres P/Q faibles pour les algorithmes EDH/DHE.
- **Où ?** : Dans `crypto/dh/dh_key.c` une partie de code génère un faux positif dans certains cas (sur une condition de test).
- **Solution :** Logiciel *Nessus Vulnerability Scanner* pour tester la configuration des serveurs.

Chiffrement et protocoles I

RSA-OAEP

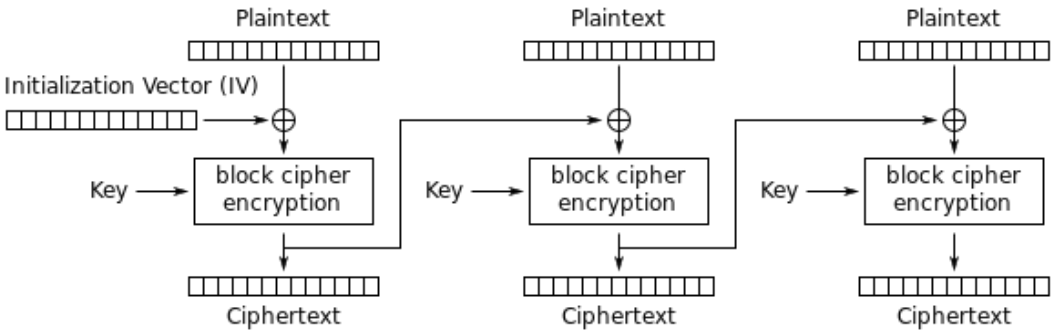
laïus RSA-OAEP (fonctionnement, +, -)

Manger's attack

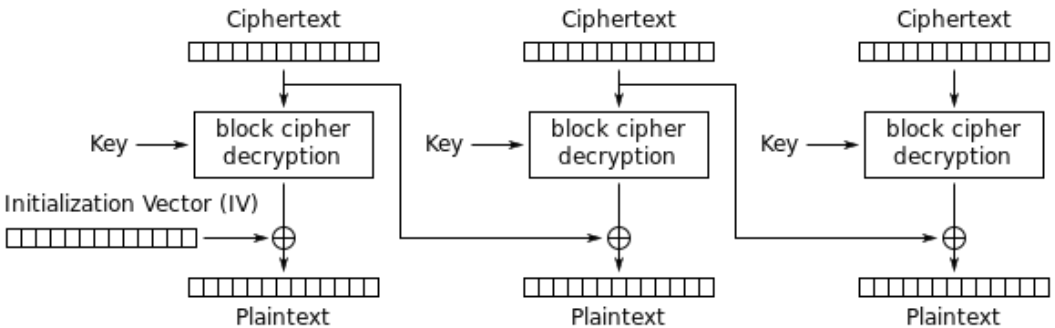
- OpenSSL 1.0.0 ;
- OAEP : défaillant ?
- contrôler la taille des paramètres à hacher ;
- sur serveur : variations de délais ;
- systèmes embarqués : plus problématique.

Chiffrement et protocoles II

CBC



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Chiffrement et protocoles III

Man in the Middle

- attaque à clair choisi ;
- récupérer cookies de session.

Problème

- SSL/TLS chiffre un canal de communication ;
- problème d'IV.

Recommandations

- ne pas utiliser CBC ;
- concaténer tous les objets.

Signature et authentification I

Définition

- Juridiquement, une signature électronique a même valeur qu'une signature manuscrite.
- Elle **DOIT** assurer l'intégrité, l'authentification et la non-répudiation d'un message.

Des anciennes versions d'OpenSSL ont des vulnérabilités au niveau de la vérification de messages signés, ou des fuites d'informations sur la clef privée ayant servi à chiffrer.

Signature et authentification II

Audit : Attaque par injection de fautes sur les certificats RSA.

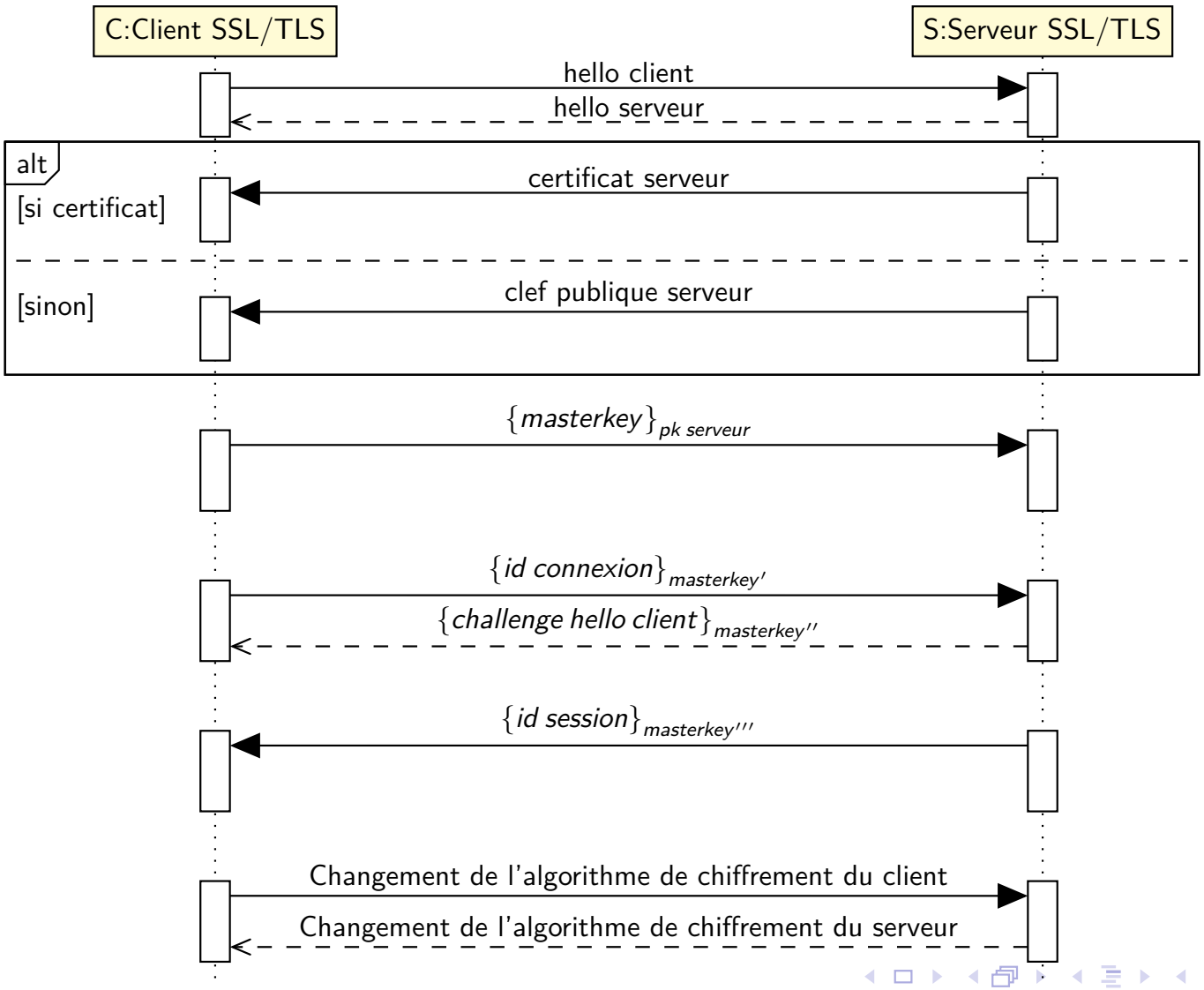
- **Description** : L'attaque se fait sur des morceaux de la signature afin de récupérer la clef privée bit à bit.
- **Comment ?** : Du bon matériel, surtout au niveau de la mémoire vive (i.e. Système Linux avec une architecture SPARC) et un oracle (e.g. système de prédictions) permettant de fabriquer la clef.
- **Temps de l'attaque** : une centaine d'heures.

Signature et authentification III

Audit : Attaque par injection de fautes sur les certificats RSA.

- **un problème dans le code OpenSSL ?** : la fonction `Fixed_Window_Exponentiation` utilise des milliers de multiplications, qui est l'opération la plus sensible en cas de dégradation du micro-processeur.
- **solution :** Aucune ! On pourrait utiliser la technique du `square_and_multiply` mais elle a l'inconvénient d'être vulnérable à une attaque par *timing*.
- **conséquences :** à moins que l'attaquant n'ait accès physiquement à votre machine les risques sont faibles. Cependant l'Université du Michigan cherche un moyen de faire des injections à distance à base d'`impulsions lasers`.

Protocole SSL/TLS I



Protocole SSL/TLS II

Historique

- SSL 1.0 et 2.0 (1995) conçues par Netscape, 3.0 (1996) par l'IETF ;
- TLS 1.0 (1999) : légère amélioration de SSL 3, ajout d'extensions ;
- TLS 1.1 (2006) : protection contre les attaques contre CBC ;
- TLS 1.2 (2008) : remplacement MD5/SHA1 par SHA256, ajout des modes GCM et CCM.

Failles notables d'OpenSSL

SSL 3 et TLS 1.0 :

- 2002 - attaque sur le padding du mode CBC par Serge Vaudenay (reprise par Lucky Thirteen) : *timing attack* ;

TLS 1.1 :

- **2011** (CVE-2011-4576) - récupération d'informations sur un échange précédent : mémoire non réinitialisée ;
- **2013** (CVE-2013-0169)- Lucky Thirteen : *timing attack*, 2^{23} sessions TLS pour retrouver un bloc en clair ;

TLS 1.2 :

- **2013** (CVE-2013-6449) - déni de service en envoyant une structure mal formée causant un *crash* dans la fonction `ssl_get_algorithm2`

Ouverture

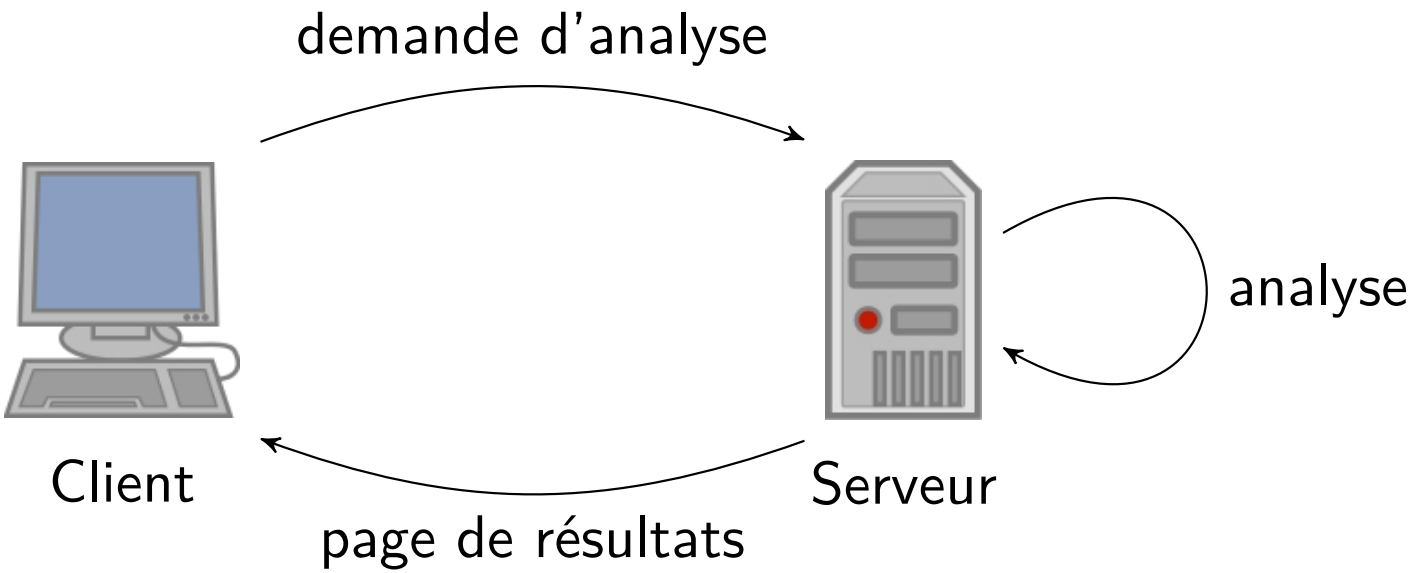
Alternatives

- CyaSSL ;
- GnuTLS ;
- MatrixSSL ;
- Network Security Services (Firefox) ;
- PolarSSL (pas de support du DTLS) ;
- Java Secure Socket Extension (pas de support du DTLS).

Attention – 25 février 2014

CVE-2014-1959 dans GnuTLS : validation de certificat (X509 version 1) intermédiaire comme un certificat d'AC par défaut.

Faiblesses identifiées I



Faiblesses identifiées II

Critères

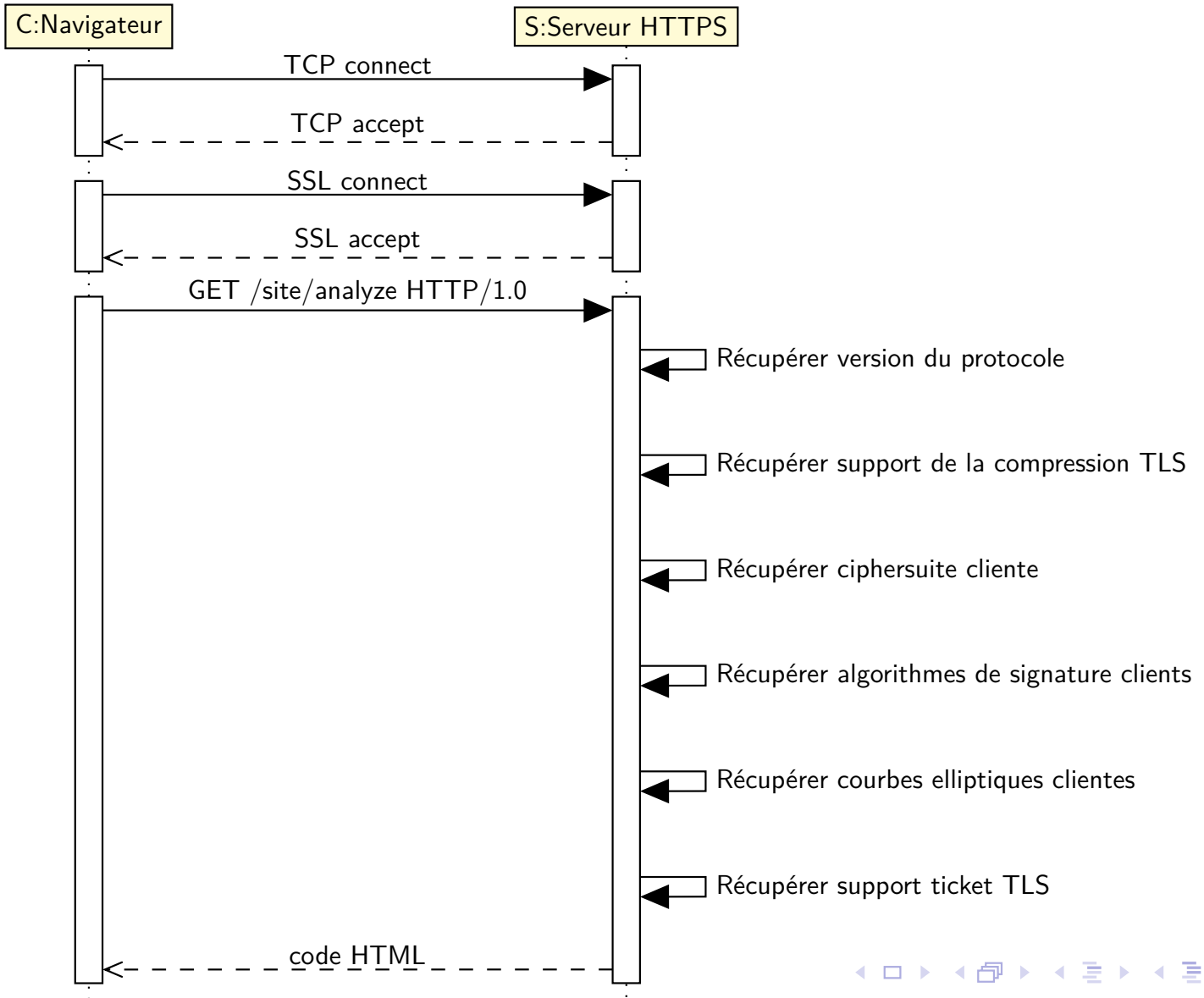
- version du protocole ;
- *ciphersuites* proposées par le client ;
- courbes elliptiques supportées ;
- algorithmes de signature ;
- compression TLS ;
- activation ou non du ticket de session.

Implémentation I

Architectures possibles

1. journalisation de la connexion HTTPS avec une sonde IDS puis récupération des informations avec un langage tel que PHP $\rightarrow$ couche réseau ;
2. serveur HTTPS avec *libssl* puis analyse de la structure de la session $\rightarrow$ couche applicative.

Implémentation II



Qualys SSL Labs

Description

- Qualys est une plateforme multi-sécurité :
- Le projet SSL Labs permet de :
  - tester l'implémentation SSL du navigateur ;
  - tester la configuration et les certificats d'un serveur ;
  - lister les bonnes pratiques sur les déploiements SSL/TLS.

Qualys SSL Labs

Analyse des serveurs

L'analyse porte sur les points suivants :

- contrôle du certificat ;
- protocoles supportés ;
- échange des clés ;
- qualité de chiffrement ;
- système de notation.

Tests sur un certificat vulnérable

SSL Report : `spXXXX.XXXX.net` (`85.XXX.XXX.33`)

- Notation : F
- Certificat non sûr → Non-confiance dans la chaîne de certification
- Sensible à une attaque de type CRIME
- Protocoles obsolètes → TLS_RSA_EXPORT_WITH_RC4_40_MD5
- Taille de clé insuffisante → RSA 1024 bits
- Re-négociation sécurisée non supportée.
- Compression TLS non-securisée
- Généré en Janvier 2014 - Expire en Janvier 2038

Démonstration

Analyse de la sécurité des navigateurs clients

- Sous le navigateur graphique `Chrome` : le plus utilisé dans le monde.
- Sous le navigateur console `lynx` : apprécié chez les développeurs.

Modification manuelle

Nous pouvons modifier la *ciphersuite* du client avec la commande `s_client`.

Conclusion

Bilan du projet

- mise en évidence des clefs faibles ;
- évaluation du code OpenSSL.

Apports du projet

- recherche approfondie de normes ;
- riche en développement et en recherche.

Questions ?