

# 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

Titre du bloc

Contenu

# Sommaire

- 1 Introduction
  - Sujet et problématique
- 2 Audit des clefs RSA des certificats
  - Récupération
    - Adresses
    - Certificats
  - Factorisation
  - Résultats
- 3 Audit d'OpenSSL
  - Entropie
  - Entropie
- Génération des clefs
- Chiffrement et protocoles
- Signature et authentification
- Protocole SSL/TLS
- Ouverture
- 4 Analyse dynamique du navigateur client
  - Faiblesses identifiées
  - Implémentation
  - Démonstration
- 5 Conclusion

# Récupération des adresses 1

## ZMAP

- open source ;
- outil de scan réseau ;
- adresses IPv4 ;
- paquets SYN sur le port 443.

# Récupération des certificats 1

## Application Récupération de Certificats

- script perl ;
- certificats SSL ;
- stocker l'ensemble des empreintes dans un dossier :
  - certificats ;
  - clefs de session.

# Récupération des certificats 2

## Algorithme

algo de récupération des certificats

# Gestion des doublons 1

## Gestion des doublons

- script perl ;
- si une empreinte de trouve déjà dans le dossier : on la stocke dans un autre dossier, celui des doublons ;
- liens symboliques.

# Gestion des doublons 2

Algorithme gestion des doublons

algo de gestion des doublons



# Factorisation

PGCD deux à deux

$N_1$

$N_2$

$N_3$

$N_4$

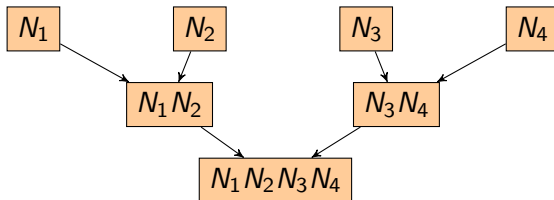
# Factorisation

## PGCD deux à deux



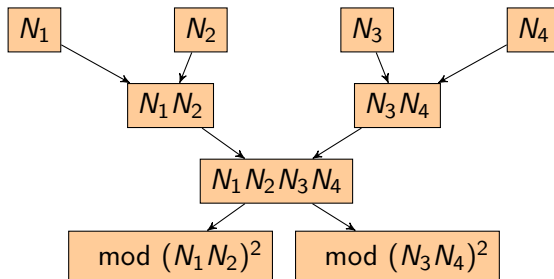
# Factorisation

## PGCD deux à deux



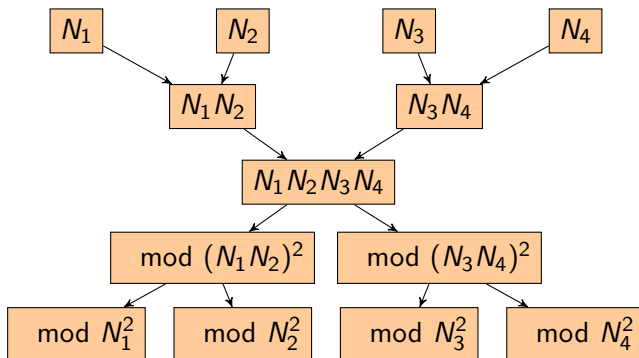
# Factorisation

## PGCD deux à deux



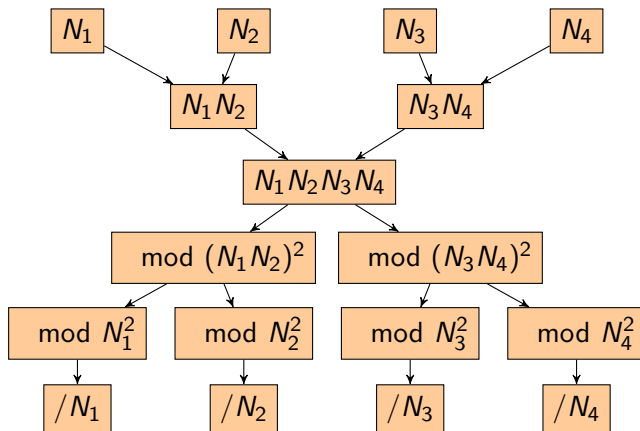
# Factorisation

## PGCD deux à deux



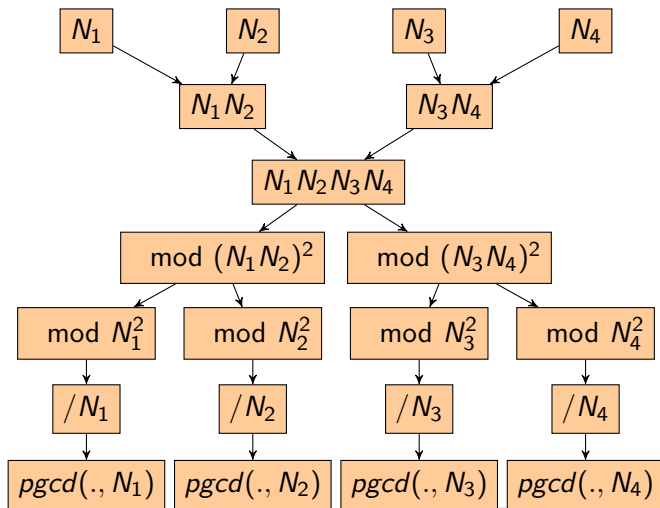
# Factorisation

## PGCD deux à deux



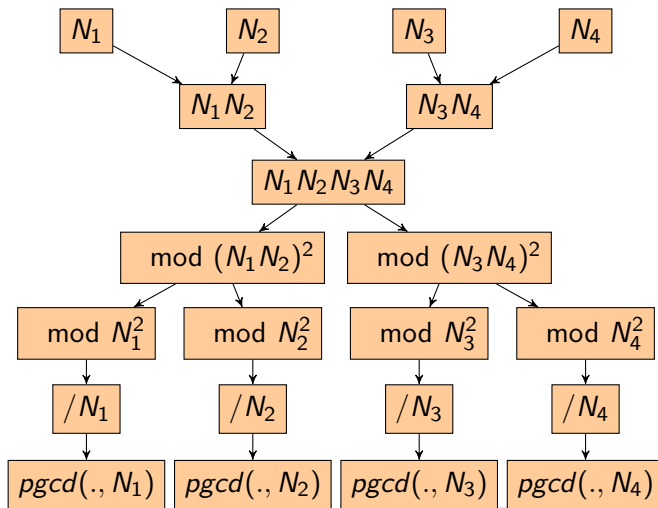
# Factorisation

## PGCD deux à deux



# Factorisation

## PGCD deux à deux





# Factorisation

## Exemple

$$2 * 3$$

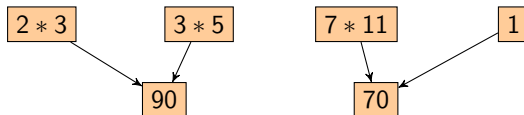
$$3 * 5$$

$$7 * 11$$

$$1$$

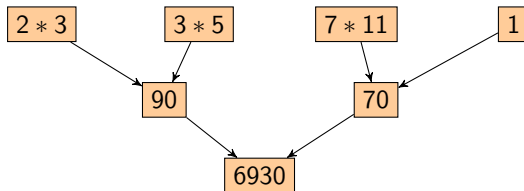
# Factorisation

## Exemple



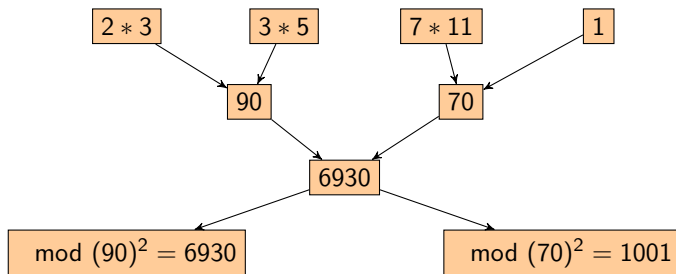
# Factorisation

## Exemple



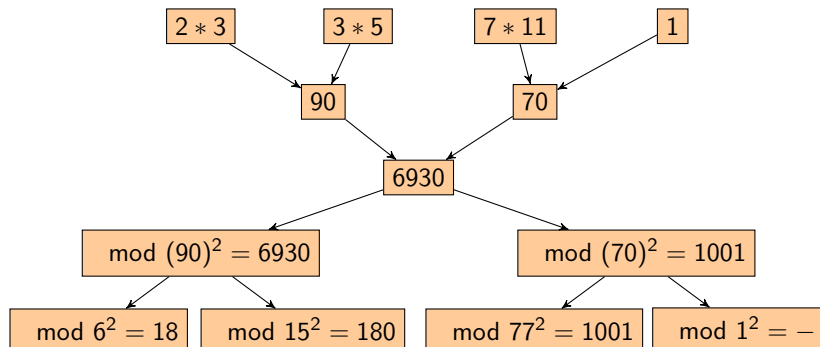
# Factorisation

## Exemple



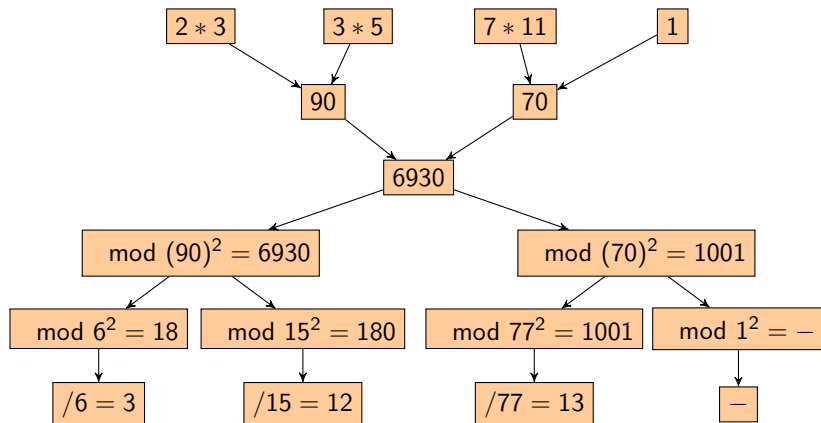
# Factorisation

## Exemple



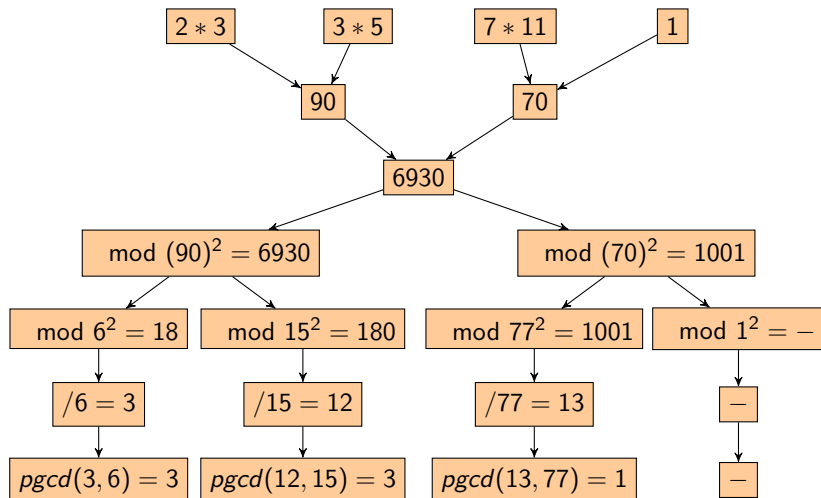
# Factorisation

## Exemple



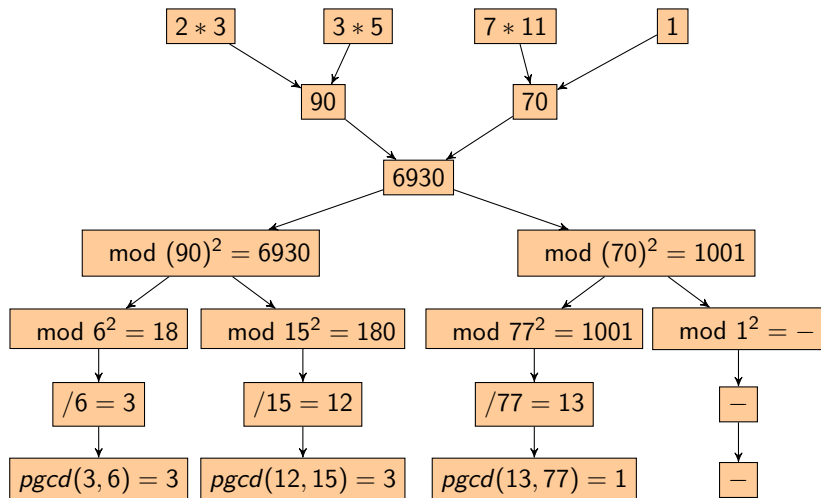
# Factorisation

## Exemple



# Factorisation

## Exemple





# Factorisation

Algorithmes associés : Arbre des produits, Arbre des restes

**Entrées:** tableau des moduli :  $T$

**Sorties:** Hauteur arbre, produits des moduli

```

 $v \leftarrow T;$ 
 $level \leftarrow 0;$ 
tant que  $|v| > 1$  faire
     $tmp \leftarrow \emptyset;$ 
    pour chaque  $i \in \{0, \dots, |v|/2\}$  faire
         $tmp[i] \leftarrow v[i \times 2] \times v[i \times 2 + 1];$ 
    fin
     $storeProductLevel(v, level);$ 
     $v \leftarrow tmp;$ 
     $level \leftarrow level + 1;$ 
fin
retourner  $level$ 

```

**Entrées:** Hauteur de l'arbre :  $level$

**Sorties:** PGCDs des moduli

```

tant que  $level > 0$  faire
     $P \leftarrow getRemainderLevel(level);$ 
     $v \leftarrow getProductLevel(level - 1);$ 
    pour chaque  $i \in \{0, \dots, |v|\}$  faire
         $v[i] \leftarrow P[i/2] \pmod{v[i]^2};$ 
    fin
     $storeRemainderLevel(v, level);$ 
     $v \leftarrow tmp;$ 
     $level \leftarrow level - 1;$ 
fin
 $w \leftarrow \emptyset;$ 
pour chaque  $i \in \{0, \dots, |v|\}$  faire
     $w[i] \leftarrow P[i/2] \pmod{v[i]^2};$ 
     $w[i] \leftarrow w[i]/v[i];$ 
     $w[i] \leftarrow pgcd(w[i], v[i]);$ 
fin
retourner  $w$ 

```

# Factorisation – Démonstration

# Résultats

# Introduction

## Contexte

- récent scandale sur la NSA ;
- beaucoup d'outils utilisés.

Mais à qui peut-on faire confiance ?

## Audit d'OpenSSL

Cinq grands axes :

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

# Entropie

## Définitions

### Générateur

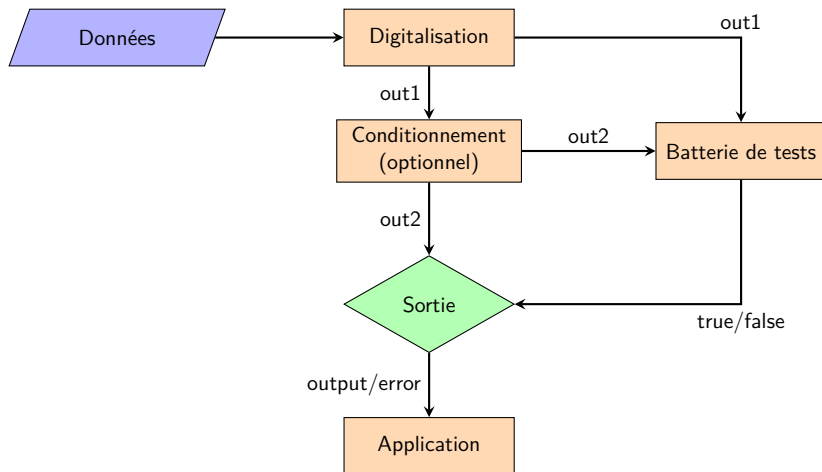
- problème aléatoire ;

- mesure de l'entropie

$$H(X) = - \sum_x P(X = x) * \log_2(P(X = x)) ;$$

- entropie et moduli.

# Entropie



# Entropie

## Tests

- Tests trop anciens, FIPS1 (1994), non mis à jour
  - test monobit
  - test poker
  - test runs
  - test long runs
- Autres tests proposés dans le rapport

# Entropie – Démonstration

## Faible 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 été 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** : Entre 2020 et 2030.



# Génération des clefs

## Principes de Kerckhoff

- Le *secret* réside dans la clef ;
- Les algorithmes de génération de clés ne doivent donner aucune information sur la clé.

## Deux grands types de générateur de clés

- Générateurs de bits aléatoires (RNG) 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

## 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

# Signature et authentification

## Définition

- Juridiquement, une signature électronique à 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 clé privée ayant servi à chiffrer.

# Signature et authentification

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

- **Description** : L'attaque se fait sur des morceaux de la signature récupéré afin de récupérer la clé 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 clé.
- **Temps de l'attaque** : une centaine d'heure.

# Signature et authentification

## 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 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** : A moins que l'attaquant n'ai 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

# Ouverture



# Faiblesses identifiées

## 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

# 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