Étude pratique de l'attaque de Bleichebacher avec Wombat

Olivier Levillain Aina Toky Rasoamanana

Télécom SudParis

Conférence ESSI
11 septembre 2019

Plan

Rappels sur RSA et PKCS#1 v1.5

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Résultats et travaux en cours

Conclusion

Plan

Rappels sur RSA et PKCS#1 v1.5

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Résultats et travaux en cours

Conclusion

Petits rappels de cryptographie sur RSA

RSA

- un cryptosystème encore très utilisé
- permet le chiffrement asymétrique et signature

Petits rappels de cryptographie sur RSA

RSA

- un cryptosystème encore très utilisé
- permet le chiffrement asymétrique et signature

Fonctionnement

- clé publique $n = pq$, e
- clé privée d
- chiffrement brut : $C = M^e[n]$
- déchiffrement brut : $C^d = M^{ed} = M[n]$

Petits rappels de cryptographie sur RSA

RSA

- un cryptosystème encore très utilisé
- permet le chiffrement asymétrique et signature

Fonctionnement

- clé publique $n = pq$, e
- clé privée d
- chiffrement brut : $C = M^e[n]$
- déchiffrement brut : $C^d = M^{ed} = M[n]$

Problèmes avec ces opérations brutes

- si e et M sont « petits »
- malléabilité du cryptosystème

Besoin d'un schéma de bourrage

Il faut donc formater le message avant de le chiffrer (ou de le signer)

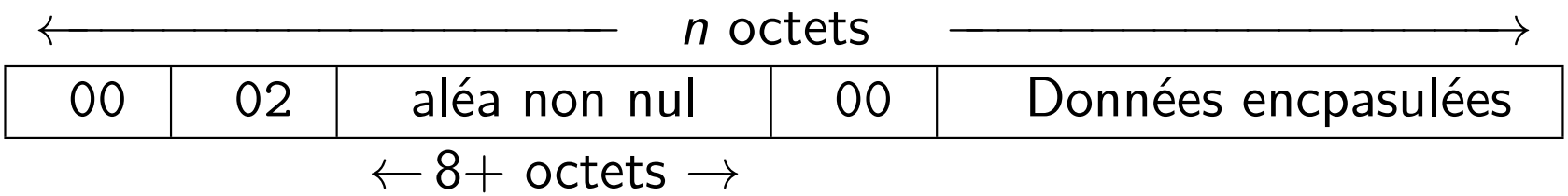
- PKCS#1 standardise la manière d'utiliser RSA
- en particulier, le document définit différents schémas de *padding*

Besoin d'un schéma de bourrage

Il faut donc formater le message avant de le chiffrer (ou de le signer)

- PKCS#1 standardise la manière d'utiliser RSA
- en particulier, le document définit différents schémas de *padding*

Le schéma qui nous intéresse est le *padding* de type 2, décrit dans la version 1.5 du standard pour le chiffrement :



Autres schémas de bourrage

PKCS#1 v1.5 décrit deux autres schémas, déterministes

- *padding* de type 0 (utilisation d'octets nuls, peu utilisé ?)
- *padding* de type 1 (octets `ff`, utilisé pour la signature)

PKCS#1 v2.1

- OAEP (Optimal Asymmetric Encryption Padding) pour le chiffrement
- PSS (Probabilistic Signature Scheme) pour la signature
- ces schémas ont de meilleures propriétés...
- ... mais ne sont pas toujours utilisées dans les standards

Plan

Rappels sur RSA et PKCS#1 v1.5

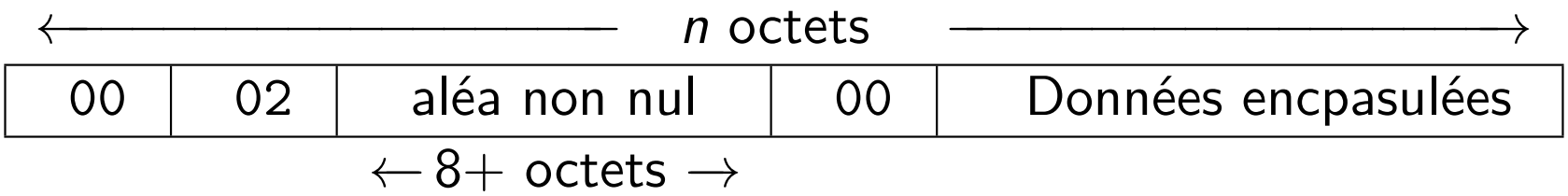
Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

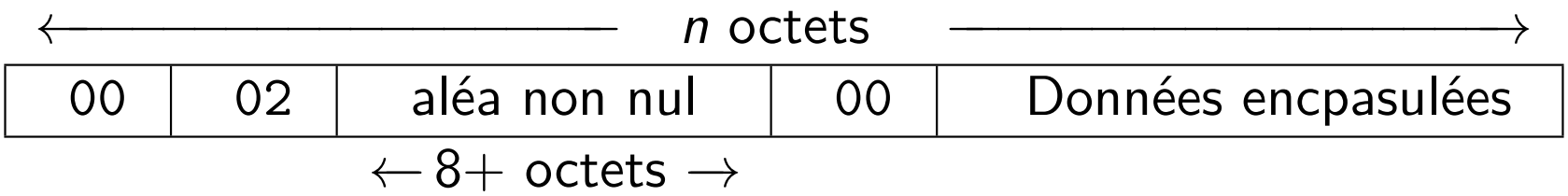
Résultats et travaux en cours

Conclusion

Un constat sur le *padding type* 2



Un constat sur le *padding type* 2



Lorsqu'un message à chiffrer est correctement formaté

- le clair « brut » M commence par les octets 00 02
- ramené à une équation mathématique, cela signifie que M est compris entre $2B$ et $3B$
  - avec $B = 2^{(|n|-16)}$
  - où $|n|$ est la taille du module en bits

Principe de l'attaque

On suppose qu'il existe un oracle qui

- accepte de déchiffrer des messages
- renvoie vrai lorsque le *padding* est bon, faux sinon
- (le message déchiffré reste secret)

Principe de l'attaque

On suppose qu'il existe un oracle qui

- accepte de déchiffrer des messages
- renvoie vrai lorsque le *padding* est bon, faux sinon
- (le message déchiffré reste secret)

Un attaquant souhaitant découvrir $m = c^d$ peut alors

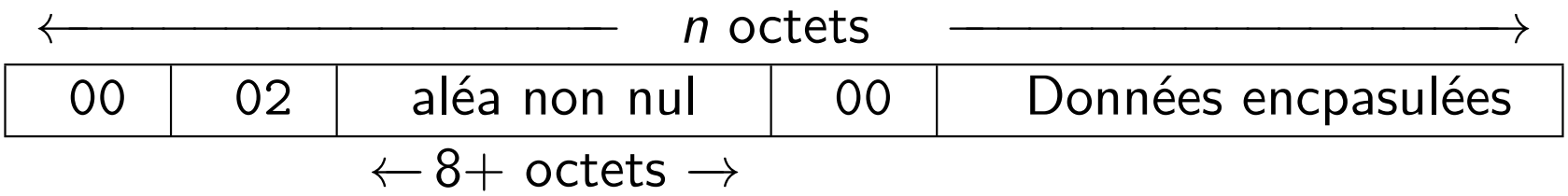
- envoyer des messages de la forme cs^e
- laisser le serveur analyser $(cs^e)^d = c^d \cdot s^{ed} = ms$
- déduire que $2B \leq ms < 3B$ si l'oracle répond vrai
- recommencer, et retrouver m grâce à une attaque à chiffré choisi adaptative

Différents types d'oracles (1/2)

En pratique, ce que l'attaquant souhaite, c'est découvrir des messages commençant par 00 02.

Différents types d'oracles (1/2)

En pratique, ce que l'attaquant souhaite, c'est découvrir des messages commençant par 00 02.



Cependant, certains oracles font d'autres vérifications

- le *padding* contient au moins 8 octets
- le *padding* se termine
- le message décapsulé a bien la taille attendu

Différents types d'oracles (2/2)

Supposons que l'oracle ne renvoie vrai que pour les messages vérifiant

- m commence par `00 02`
- le *padding* contient au moins 8 octets
- le *padding* se termine

L'attaquant perd alors des « bons » messages (commençant par `00 02`) qui lui auraient donné de l'information.

Différents types d'oracles (2/2)

Supposons que l'oracle ne renvoie vrai que pour les messages vérifiant

- m commence par `00 02`
- le *padding* contient au moins 8 octets
- le *padding* se termine

L'attaquant perd alors des « bons » messages (commençant par `00 02`) qui lui auraient donné de l'information.

Dans Bardou et al., une classification est proposée, selon les types de messages que l'attaquant peut distinguer grâce à l'oracle.

Résultats de Bardou et al.

L'article publié à CRYPTO 2012 propose des améliorations à l'algorithme original (CRYPTO 1998)

Type d'oracle	**Nb moyen de requêtes**	
	Algo original	**Algo optimisé**
FFF	-	18 040 221
FTT	215 982	49 001
FTT	159 334	39 649
TFT	39 536	10 295
TTT	38 625	9 374

Plan

Rappels sur RSA et PKCS#1 v1.5

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Résultats et travaux en cours

Conclusion

Implémentation de l'attaque de manière modulaire (1/2)

Pour tester une implémentation, on crée un *stub*, qui permet de

- récupérer la clé publique
- obtenir un chiffré à attaquer (le *challenge*)
- soumettre des messages à déchiffrer

Implémentation de l'attaque de manière modulaire (1/2)

Pour tester une implémentation, on crée un *stub*, qui permet de

- récupérer la clé publique
- obtenir un chiffré à attaquer (le *challenge*)
- soumettre des messages à déchiffrer

L'attaquant soumet des messages particuliers pour tester le type d'oracle

- messages bien formés
- messages ne commençant pas par 00 02
- messages avec un padding trop court
- messages sans message à décapsuler

Implémentation de l'attaque de manière modulaire (2/2)

Si l'attaquant sait identifier des « bons » messages à l'aide des réactions observées, un oracle est identifié

Il peut alors

- évaluer de manière plus fine le coût de l'attaque
- réaliser l'attaque pour retrouver le clair correspondant au *challenge*
- utiliser l'oracle pour produire une signature

Implémentation de l'attaque de manière modulaire (2/2)

Si l'attaquant sait identifier des « bons » messages à l'aide des réactions observées, un oracle est identifié

Il peut alors

- évaluer de manière plus fine le coût de l'attaque
- réaliser l'attaque pour retrouver le clair correspondant au *challenge*
- utiliser l'oracle pour produire une signature

Pour cela, `wombat` implémente actuellement

- l'attaque originale de Daniel Bleichenbacher
- des versions améliorées de l'attaques (Bardou et al.)
- des oracles purs pour tester ces attaques
- un *stub* TLS (voir plus loin)

Outil développé et publié sous licence libre

Wombat

- outil développé en python dans le cadre d'un stage
- publication de la version 0.1 début septembre
- `https://gitlab.com/pictyeye/wombat`

Outil développé et publié sous licence libre

Wombat

- outil développé en python dans le cadre d'un stage
- publication de la version 0.1 début septembre
- `https://gitlab.com/pictyeye/wombat`

Utilisations possibles

- faciliter les tests pour identifier la présence d'oracles
- mener des attaques dans le cadre de la loi et le respect de la morale
- monter des TP (il existe un exemple de serveur simple vulnérable)

Plan

Rappels sur RSA et PKCS#1 v1.5

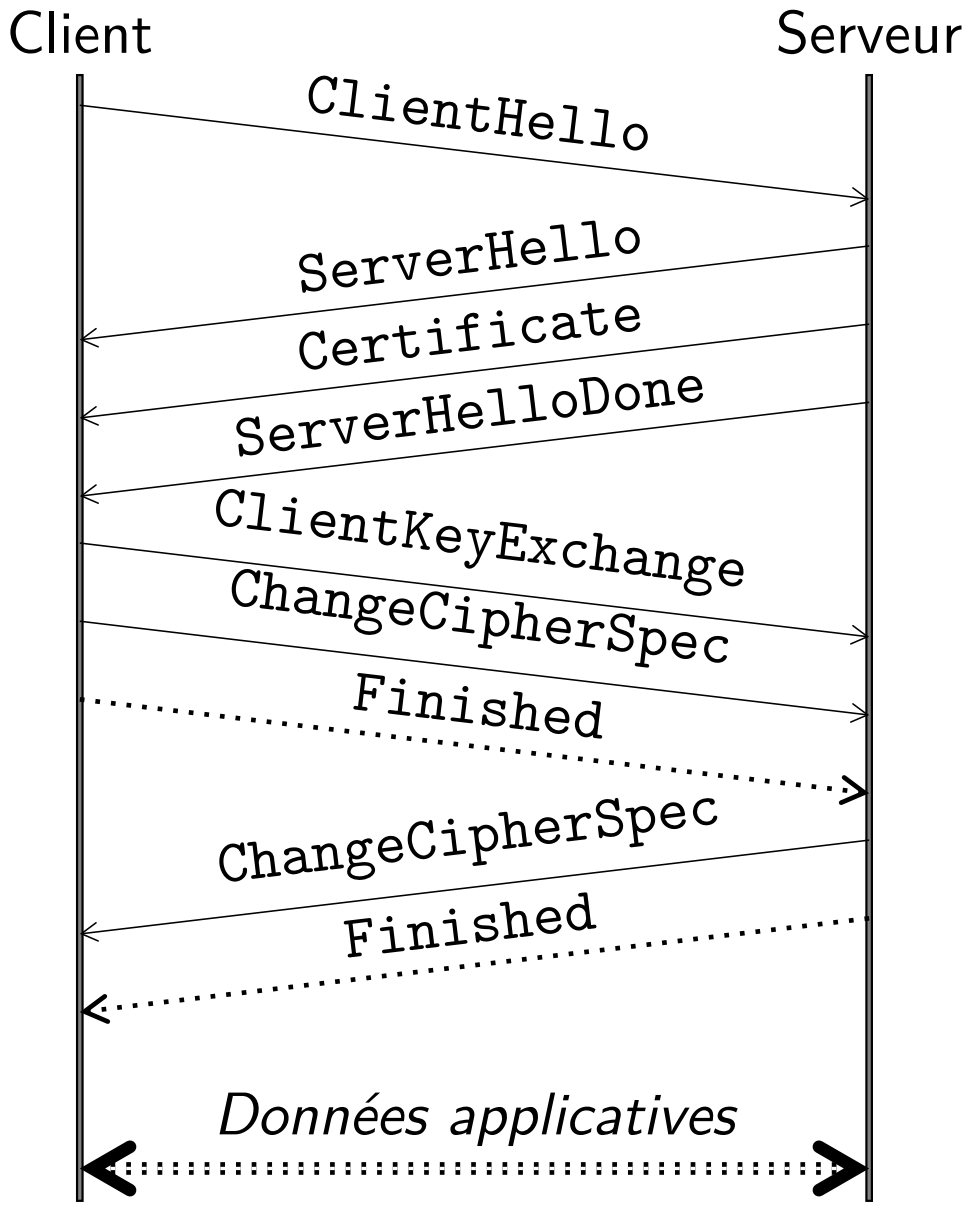
Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

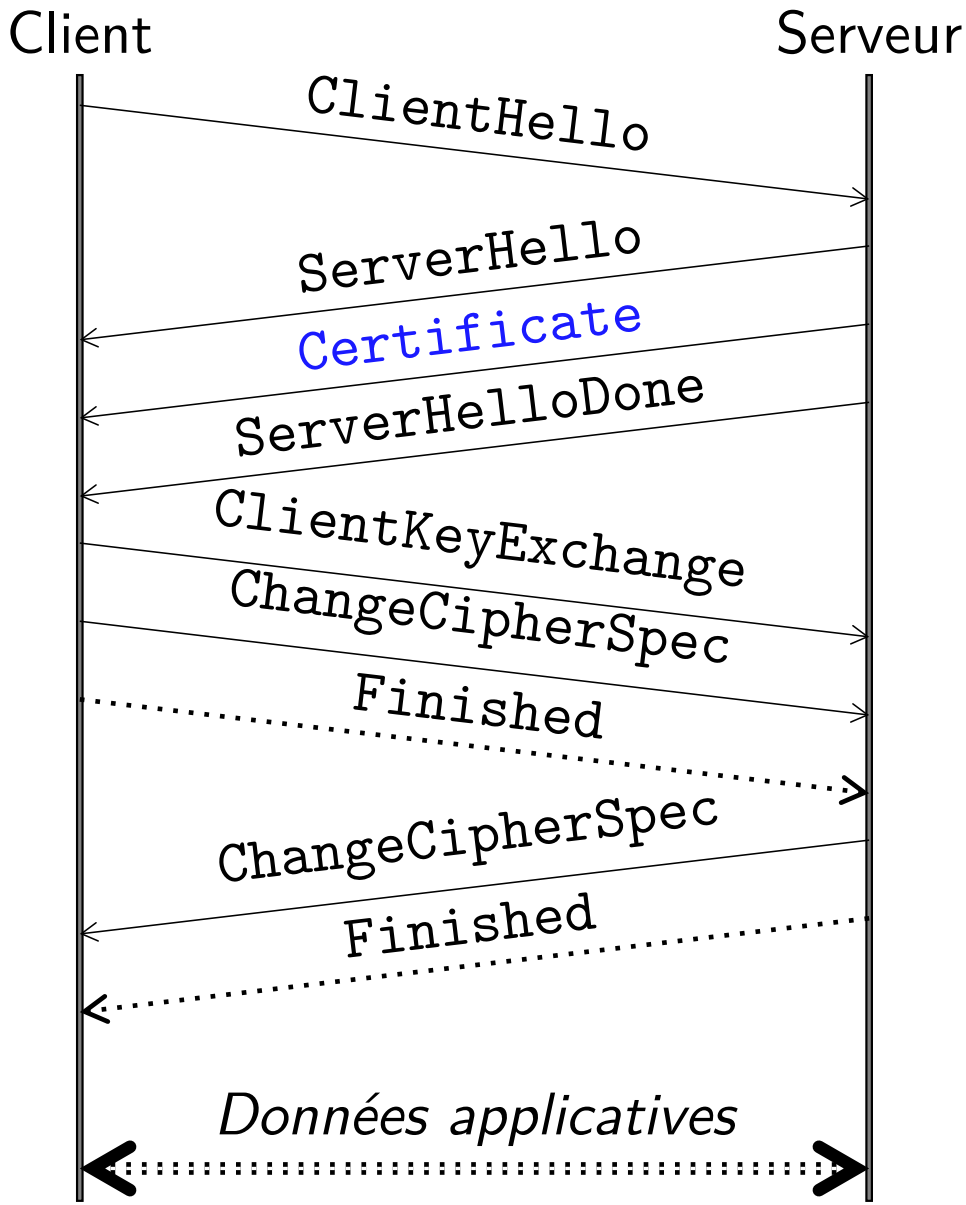
Résultats et travaux en cours

Conclusion

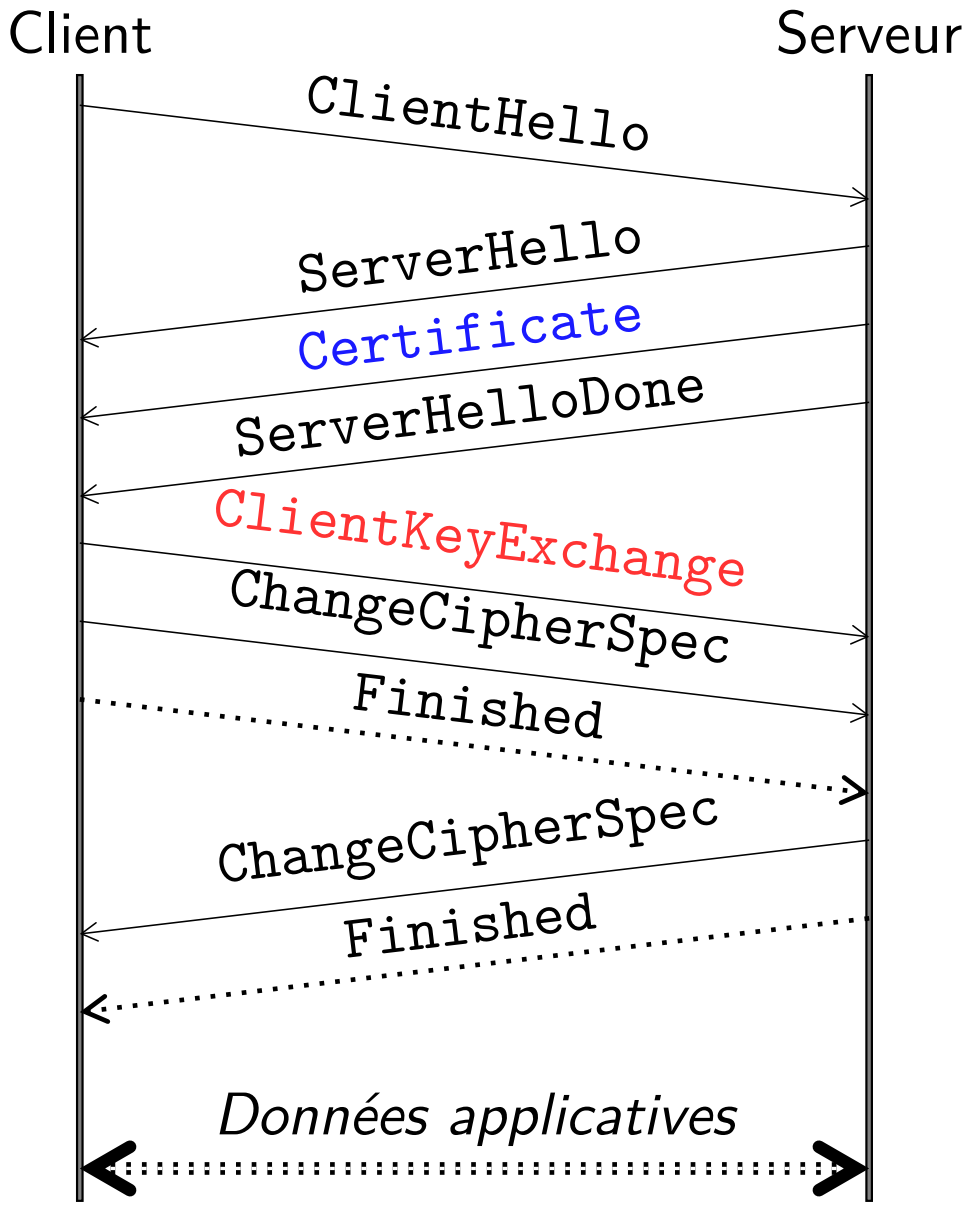
Application à TLS (1/3)



Application à TLS (1/3)



Application à TLS (1/3)



Application à TLS (2/3)

Création d'un *stub*

- récupération de la clé publique à partir du certificat du serveur
- production d'un chiffré contenant un *pre-master-secret*
- demande de déchiffrement au serveur TLS
  - réalisation du premier échange en forçant l'échange de clé RSA
  - envoi du `ClientKeyExchange` qui contient le message chiffré à tester
  - observation de la réaction du serveur (messages, temps de traitement)

Identification et exploitation d'un éventuel oracle

Application à TLS (3/3)

Ajout d'une vulnérabilité à `mbedtls` (l'implémentation officielle est très robuste, y compris pour les *timing attacks*)

```
+   // DON'T DO THIS AT HOME
+   if (ret == MBEDTLS_ERR_RSA_INVALID_PADDING) {
+       mbedtls_ssl_send_alert_message( ssl, MBEDTLS_SSL_ALERT_LEVEL_FATAL,
+                                       MBEDTLS_SSL_ALERT_MSG_DECRYPT_ERROR );
+       return (MBEDTLS_ERR_SSL_BAD_HS_CLIENT_KEY_EXCHANGE);
+   }
```

Application à TLS (3/3)

Ajout d'une vulnérabilité à `mbedtls` (l'implémentation officielle est très robuste, y compris pour les *timing attacks*)

```
+   // DON'T DO THIS AT HOME
+   if (ret == MBEDTLS_ERR_RSA_INVALID_PADDING) {
+       mbedtls_ssl_send_alert_message( ssl, MBEDTLS_SSL_ALERT_LEVEL_FATAL,
+                                       MBEDTLS_SSL_ALERT_MSG_DECRYPT_ERROR );
+       return (MBEDTLS_ERR_SSL_BAD_HS_CLIENT_KEY_EXCHANGE);
+   }
```

Démonstration

Travaux à venir sur TLS

Fiabiliser l'outil pour tester de vrais serveurs TLS

- amélioration de la « séparation » des distributions des temps de traitement
- meilleure prise en compte de tous les messages possibles (y compris des signaux au niveau TCP)
- adaptation de la stimulation par l'envoi optionnel de certains messages (ROBOT)

Travaux à venir sur TLS

Fiabiliser l'outil pour tester de vrais serveurs TLS

- amélioration de la « séparation » des distributions des temps de traitement
- meilleure prise en compte de tous les messages possibles (y compris des signaux au niveau TCP)
- adaptation de la stimulation par l'envoi optionnel de certains messages (ROBOT)

Idées de campagnes de mesures pour *identifier* d'éventuels oracles

- Top Alexa 1M (*spoiler alert* : il existe des serveurs vulnérables)
- serveurs SMTP/IMAP (souvent à la traîne pour leurs piles TLS)

Travaux à venir sur TLS

Fiabiliser l'outil pour tester de vrais serveurs TLS

- amélioration de la « séparation » des distributions des temps de traitement
- meilleure prise en compte de tous les messages possibles (y compris des signaux au niveau TCP)
- adaptation de la stimulation par l'envoi optionnel de certains messages (ROBOT)

Idées de campagnes de mesures pour *identifier* d'éventuels oracles

- Top Alexa 1M (*spoiler alert* : il existe des serveurs vulnérables)
- serveurs SMTP/IMAP (souvent à la traîne pour leurs piles TLS)

Implémenter des attaques plus sophistiquées (DROWN, signature TLS 1.3)

Autres applications

PKCS#1 est présent dans d'autres standards

- XML Encryption
- SSH (RFC 4432)
- OpenPGP

Autres applications

PKCS#1 est présent dans d'autres standards

- XML Encryption (projet proposé aux étudiants de TSP)
- SSH (RFC 4432)
- OpenPGP

Autres applications

PKCS#1 est présent dans d'autres standards

- XML Encryption (projet proposé aux étudiants de TSP)
- ~~SSH~~ (la RFC 4432 utilise OAEP)
- OpenPGP

Autres applications

PKCS#1 est présent dans d'autres standards

- XML Encryption (projet proposé aux étudiants de TSP)
- ~~SSH~~ (la RFC 4432 utilise OAEP)
- OpenPGP (investigations en cours)

Plan

Rappels sur RSA et PKCS#1 v1.5

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Résultats et travaux en cours

Conclusion

Conclusion

La *million-message attack* de Bleichenbacher

- une attaque connue depuis longtemps
- un exemple non trivial pour former les étudiants en sécurité
- encore une réalité aujourd'hui ?

Wombat

- un outil libre pour tester l'attaque
- possibilité de reproduire des attaques existantes
- extension via des *stubs* pour scruter d'autres standards
- utilisation possible en TP

Questions ?

Merci pour votre attention

`https://gitlab.com/pictyeye/wombat`