Cryptologie CRYPA: PROJET(s)

robert.erra@epita.fr

Cours CRYPTO ING3 2017:
Projet obligatoire et **individuel!**
Pour toute question: merci d'utiliser
cryptoing3@gmail.com

Examen, projet, etc.

- Pas d'examen ? Un projet individuel
- Choisissez votre projet ? (voir ensuite deux exemples)
- Sous quelle forme ? un seul fichier, pas de limite de pages (mais restez raisonnables svp, je lis tout)
- Métriques : au moins 5000 caractères et pas plus de 15000. Hors images donc et HORS CODE.
- Comment ? envoyer votre fichier à : **cryptoing3@gmail.com** avec le nom « dedans » le mail et « dedans » le fichier SVP !
- Quand ? avant le vendredi 15 décembre 23h42.
- Option : Si vous ne voulez prendre ni le projet 1 ni le projet 2, et que par exemple que vous désirez prendre un sujet en rapport avec votre poste en entreprise, envoyez moi un message **clair** avec un bref résumé. Et vous devez avoir reçu une confirmation claire de ma part avant de considérer que c'est accepté. (Exemple : remplacer RSA par DH.)

Projets

Sujet du projet 1

- Sujet ? Choisir une bibliothèque « open source » crypto, dans le langage qui vous plait le plus, ou suivant tout autre critère de choix, différentes de celle présentées dans les transparents plus loin [ou alors prenez la dernière version].
- Regardez comment sont calculés les nombres premiers : quels tests pour éliminer rapidement les nombres composés (si de tel tests sont présents) ?
- Regardez comment est généré le « random » initial. (Exemple, s'il y a une fonction *random*, précisez d'où elle vient et en gros comment elle fonctionne ?).
- Traduire cela en « pseudo c » (voir plus loin un exemple tiré des transparents du cours)
- Identifier proprement les éléments. Et décrire l'enchainement précis des calculs.
- Conclure sur le niveau de sécurité

Sujet du projet 2

- Sujet ? Choisir une bibliothèque « open source » crypto, dans le langage qui vous plait le plus, ou suivant tout autre critère de choix, différentes de celle présentées dans les transparents plu loin [ou alors prenez la dernière version].
- Regardez comment sont réllement calculés : p, q, n, e, d (et surtout : dans quel ordre)
- Traduire cela en « pseudo c » (voir plus loin des exemples tiré des transparents du cours)
- Identifier proprement les éléments. Et décrire l'enchainement précis des calculs.
- Exemple 1 : s'il y a une fonction *random*, précisez d'où elle vient et en gros comment elle fonctionne ?
- Exemple 2 : Calcule t-on e avant de calculer p et q ou le contraire ?
- Conclure sur le niveau de sécurité

— Algorithm used by GnuPG v1.2.3 to compute e *after* the computation of p and q.

Algorithme 1 : Computation of e

...
If $\varphi(N) \neq 0 \bmod [41]$ **Then** $e = 41$;
Else If $\varphi(N) \neq 0 \bmod [257]$ **Then** $e = 257$;
Else
$e = 65537$;
While $GCD(e, \varphi(N)) \neq 1 : e = e + 2$;

— RSA in GnuPG v1.4.10 : $e \geq 65537$

Algorithme 2 : RSA key generation

Input : — an integer k > 0 ;

Output : — (N, e, d) with N a k bit number

Begin :

$e = 65537$;

While $bitSize(N) \neq k$

Compute randomly a prime p of $k/2$ bits ;

Compute randomly a prime q of $k/2$ bits ;

Compute $N = p\,q$ and $\varphi(N) = (p-1)(q-1)$;

While $GCD(e, \varphi(N)) \neq 1$ $e = e + 2$;

/* Again : if $e > 65537$, we gain information about φ(N) */

Compute $d = e^{-1} \bmod \varphi(N)$;

End.

— RSA in libgcrypt 1.4.4 : $e \geq 65537$

Algorithme 3 : RSA key generation (it follows ANS X9.31)

Input : — an integer k = 1024 + 256s > 0 ;

Output : — (N, e, d) with N a k bit number

Begin :

$e = 65537$;

Compute randomly a prime p of $k/2$ bits ;

Compute randomly a prime q of $k/2$ bits ;

Compute $N = p\,q$ and $\varphi(N) = (p-1)(q-1)$;

Compute $\lambda(N) = lcm(p-1, q-1) = \varphi(N)/gcd(p-1, q-1)$

While $GCD(e, \lambda(N)) \neq 1$ $e = e + 2$;

/* Again : if $e > 65537$, we gain information about $\lambda(N)$ */

Compute $d = e^{-1} \bmod f$;

End.

Can we do better ? Yes . . .

— RSA in OpenSSL 0.9.8k

Algorithme 4 : RSA key generation

Input : — an integer k ;

Output : — (N, e, d, d_p, d_q) with N a k bit number

Begin :

$e = 65537$; /* e is fixed, p and q are recomputable */

While $gcd(e, p-1) \neq 1$ Compute rand. prime p of $k/2$ bits ;

While $gcd(e, q-1) \neq 1$ Compute rand. prime q of $k/2$ bits ;

Compute $N = p\,q$ and $\varphi(N) = (p-1)(q-1)$;

Compute $d = e^{-1} \bmod \varphi(N)$;

Compute $d_p = d \bmod (p-1)$;

Compute $d_q = d \bmod (q-1)$;

End.