Calcul sécurisé – Feuille d'exercices numéro 2

Université Paris-Saclay – M1 Informatique – Site de Versailles

27 février 2020

Exercice 1 (*Bug Attack*)
On suppose dans cet exercice qu'on utilise un microprocesseur 32 bits dans lequel l'opération de multiplication (notée MULT) est "buggée". Cela siginifie qu'il existe deux valeurs a_0 et b_0 de 32 bits, telles que :

$$\forall (a, b) \neq (a_0, b_0), \mathrm{MULT}(a, b) = a \times b$$

mais

$$\mathrm{MULT}(a_0, b_0) \neq a_0 \times b_0$$

1. On utilise une implémentation de RSA avec CRT (*Chinese Remainder Theorem*) et "square and multiply", qui utilise cette multiplication MULT. On suppose que l'attaquant peut lancer le calcul de $x^d \bmod n$ avec les valeurs de x qu'il souhaite. Par ailleurs on suppose qu'il connaît les valeurs de a_0 et b_0. Montrer comment il peut retrouver la clé secrète du RSA.

2. Dans la question précédente, on a fait l'hypothèse que l'attaquant connaissait a_0 et b_0. Décrire un scénario réaliste où cela peut se produire. Discuter.

Exercice 2 (*Attaque par faute contre un algorithme de chiffrement par blocs*)
Le but de cet exercice est de montrer qu'introduire des fautes dans un algorithme de chiffrement par blocs peut avoir un effet important sur sa sécurité. Tout au long de l'exercice, on considère un algorithme de chiffrement par blocs noté E avec ℓ tours, une taille de blocs et une taille de clé de n bits. Cet algorithme de chiffrement par blocs consiste simplement en l'itération de fonctions T_i et de XOR avec des sous-clés (voir Figure 1). Les sous-clés k_i, $0 \leq i \leq \ell$, sont toutes dérivées de la clé k associée à E. Le i-ième tour est noté R_i, et l'état intermédiaire du clair p après le i-ième tour est noté p_i. On peut donc écrire :

$$R_0(p) = k_0 \oplus p = p_0$$

$$R_i(p_{i-1}) = T_i(p_{i-1}) \oplus k_i = p_i \quad \text{pour } 1 \leq i \leq \ell$$

et le chiffré est $c = p_\ell$.

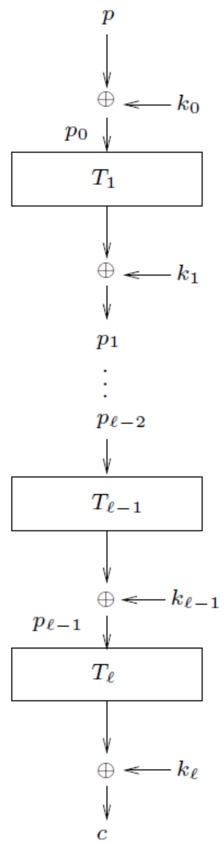

Figure 1

1. Montrer comment fonctionne le déchiffrement. Sous quelle condition peut-on déchiffrer les messages qui ont été chiffrés par E ?

À partir de maintenant, on suppose qu'on possède un dispositif permettant de provoquer des fautes dans une implémentation donnée de E (par exemple dans une carte à puce). De façon usuelle, une faute consistera à inverser un bit choisi dans un état intermédiaire p_i. Par ailleurs, on supposera que k_i est uniformément distribué dans $\{0,1\}^n$ et que $T_1 = T_2 = \ldots = T_\ell = T$.

2. Dans cette question, on produit des fautes sur $p_{\ell-1}$, *i.e* on change $p_{\ell-1}$ en $p'_{\ell-1} = p_{\ell-1} \oplus \delta$, où δ est une suite de bits de longueur n contenant des 1 aux positions où se produisent les fautes, et 0 ailleurs. Soit c' le chiffré obtenu lorsqu'on introduit les fautes δ. Trouver une relation entre δ, $p_{\ell-1}$, c et c'.

3. Supposons dans cette question que notre dispositif ne nous permet d'introduire des fautes que dans les sous-clés. Comment peut-on obtenir le même c' que ci-dessus avec un tel dispositif ?

4. On suppose dans cette question que $n = 12$ et que T est définie par

$$T : (x_1, x_2, x_3, x_4) \mapsto ((f(x_1), f(x_2), f(x_3), f(x_4)),$$

où la fonction $f : \{0,1\}^3 \to \{0,1\}^3$ est définie par la Table 1. Maintenant, on essaie d'obtenir des informations sur une des sous-clés. Pour cela, on chiffre d'abord un clair p choisi uniformément au hasard, en utilisant l'implémentation cible de E. Puis, on chiffre le même clair à nouveau, mais cette fois en introduisant des fautes dans $p_{\ell-1}$, qui est ainsi modifié en $p_{\ell-1} \oplus \delta$, avec $\delta = (001, 000, 000, 000)$, *i.e.* on inverse le dernier bit de x_1. Soit c le chiffré $E(p)$ et c' le chiffré obtenu lorsqu'il y a une faute. Montrer qu'on peut déduire des informations sur $p_{\ell-1}$ dans le cas où $c = (110, 110, 010, 011)$ et $c' = (100, 110, 010, 011)$. Combien de candidats cela fournit-il pour la valeur de $p_{\ell-1}$?

Table 1: Definition of the function f

x	000	001	010	011	100	101	110	111
$f(x)$	101	100	010	111	110	000	001	011

5. Combien de candidats pour la valeur de k_ℓ cela donne-t-il ?

6. Soit c, c' et δ comme ci-dessus. Soit $\delta' = c \oplus c'$. Calculer $\mathrm{DP}^T(\delta, \delta')$ pour la transformation définie ci-dessus, où

$$\mathrm{DP}^T(\delta, \delta') := \mathrm{Pr}_{X \in \{0,1\}^{12}}[T(X) \oplus T(X \oplus \delta) = \delta']$$

7. On considère maintenant que n, T et δ sont redevenus arbitraires. On répète la même expérience. Soit N_ℓ le nombre de candidats restants pour k_ℓ après l'expérience. Donner une expression de N_ℓ en fonction de δ, $\delta' = c \oplus c'$, n et T.

8. Montrer que $N_\ell \geq 2$.

9. En pratique, il est très difficile de produire des fautes à une position choisie de bit. On considère à nouveau l'expérience de la question 4 sauf que l'on produit une faute pour laquelle la position du bit est uniformément distribuée au hasard, *i.e.* δ est pris uniformément au hasard parmi les chaînes de bits de taille n ayant un poids de Hamming égal à 1. On suppose également que $n = 12$ et que T est la transformation définie à la question 4. Les résultats de l'expérience fournissent $c = (101, 111, 010, 100)$ et $c' = (101, 111, 110, 100)$. Combien de candidats trouve-t-on pour la valeur de k_ℓ ?

Exercice 3 (*Attaque par faute contre DES*)
Décrire précisément une attaque par fautes contre le DES. On supposera que l'attaquant est capable d'effectuer une faute sur la valeur de sortie R_{15} du 15ème tour.

Exercice 4 (*Attaque par faute contre AES*)
Décrire précisément une attaque par fautes contre l'AES, en faisant des hypothèse sur de modèle de faute.

Exercice 5 (*Contre-mesure de Shamir pour le RSA*)
On suppose que dans une implémentation RSA (calcul de $y = x^d \bmod n$ avec $n = p \times q$), on ne dispose pas de la valeur de e, ce qui empêche d'effectuer la vérification $y^e = x \bmod n$.
Shamir a proposé la technique suivante : effectuer le calcul comme dans le cas de l'utilisation des restes chinois (CRT), mais en faisant les calculs respectivement modulo $p \times r$ et modulo $q \times r$ (au lieu de les faire modulo p et modulo q), où r est un nombre premier choisi aléatoirement.

1. Montrer comment ce calcul permet (comme dans le cas des restes chinois) d'obtenir le résultat $y = x^d \bmod n$.
2. Illustrer l'attaque par faute dans le cas de cette méthode.
3. Que peut-on ajouter pour que l'algorithme puisse détecter qu'une faute s'est produite ?
4. Quelle est la taille minimum nécessaire pour r ? Discuter en fonction de la sécurité obtenue, et donner l'impact sur le temps de calcul.