CC BY NC SA

Janvier 2024

Le digicode : attaque par canal auxiliaire

Dans cette activité, les élèves doivent utiliser les indices que laisse fuiter un digicode d'un type un peu particulier pour découvrir efficacement le code permettant d'ouvrir une porte. Ce faisant, ils jouent le rôle d'un adversaire menant une attaque de sécurité dite par canal auxiliaire.

- 20 à 30 minutes.
- À partir du cycle 4.
- Élèves en binômes.
- Objectif : Découvrir le concept d'attaque par canal auxiliaire.
- Compétences travaillées :
  - Dénombrement ;
  - Culture numérique : attaque par canal auxiliaire.

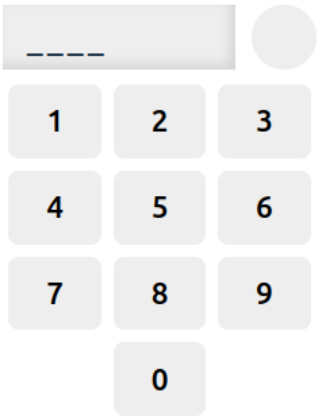

FIGURE 1 – Le digicode

1 Attaque par canal auxiliaire

En cryptographie, un canal auxiliaire (side channel en anglais) est un moyen détourné permettant d'obtenir des informations secrètes. Un exemple réel concerne les capteurs de température Netatmo qui indiquent, par défaut, publiquement sur Internet, la température observée là où ils sont placés. En hiver, il est donc possible de déterminer qu'une maison est inhabitée simplement en observant ces valeurs. Dans un cadre informatique, la mesure de la consommation électrique d'un processeur, ou celle du temps de calcul peuvent permettre d'obtenir des informations sur les calculs effectués.

FIGURE 2 – Paul Kocher

Initialement, les attaques par canal auxiliaire ont été imaginées par Paul Kocher en 1996 (voir [1]) lorsqu'il a montré qu'il était possible de découvrir les clés secrètes des chiffrements DES et RSA. Par exemple, dans la méthode RSA, le déchiffrement consiste essentiellement à effectuer un calcul d'élévation à une puissance, dans lequel l'exposant est la clé privée du destinataire. Comme les nombres en jeu sont très grands (plusieurs centaines de chiffres), un algorithme particulier, appelé l'exponentielle rapide, est utilisé pour aller vite. Cet algorithme effectue successivement des élévations au carré et des multiplications, en fonction des valeurs des bits de l'écriture en binaire de la clé privée. Comme un processeur arithmétique utilise très légèrement moins d'électricité lorsqu'il effectue une élévation au carré que lorsqu'il effectue une multiplication, l'observation de sa consommation permet d'identifier la nature des opérations effectuées, et in fine de déduire l'ensemble des bits de la clé privée. Il est possible de se prémunir contre cette attaque en modifiant légèrement l'algorithme pour qu'il effectue des opérations inutiles pour camoufler les différences engendrées par les 0 et les 1 de la clé. Une telle contre-mesure n'est pourtant pas la fin de l'histoire, puisque d'autres types d'attaques existent, en observant par exemple les défauts des mémoires caches d'un ordinateur...

2 Activité pour les élèves : l'attaque du digicode

L'objectif de cette activité est de faire découvrir le concept d'attaque par canal auxiliaire en imaginant un digicode dont les voyants lumineux laissent fuiter un peu d'information sur les chiffres du code d'entrée d'une porte.

Le code comporte 4 chiffres, et le digicode possède dix touches portant les chiffres de 0 à 9 et un voyant lumineux pouvant s'éclairer de deux couleurs : rouge et vert. Quand un chiffre correct est saisi, le voyant s'éclaire en vert, mais sitôt qu'un chiffre incorrect est tapé, le voyant s'éclaire en rouge et il faut recommencer à taper le code depuis le début.

Par exemple, si le code est 1234 et qu'Alice tape par inadvertance 124, les couleurs du voyant sont les suivantes :

- 1 $\rightsquigarrow$ le voyant s'éclaire en vert
- 2 $\rightsquigarrow$ le voyant reste éclairé en vert
- 4 $\rightsquigarrow$ le voyant s'éclaire en rouge et le code est annulé

À l'adresse `https://sancy.iut.uca.fr/~lafourcade/naive-digicode/` se trouve une simulation de ce digicode, qui permet d'expérimenter son fonctionnement.

La question posée aux élèves est : **quel est le nombre maximal d'essais qu'il faut effectuer pour ouvrir la porte sans connaître le code, pour une personne astucieuse ?**

Pour commencer, il est possible de mettre les élèves en activité en leur faisant jouer un jeu de rôle

en binômes. Un élève choisit un code secret à 4 chiffres qu'il mémorise, pour jouer ensuite le rôle du digicode, pendant que l'autre élève essaie de le deviner en indiquant les touches qu'il frappe sur le clavier. Autrement dit, à chaque saisie d'un chiffre sur le clavier, l'élève qui joue le rôle du digicode indique seulement la couleur du voyant. Puis les deux rôles peuvent être permutés.

Pendant cette activité, les élèves trouvent rapidement les codes secrets : le nombre d'essais nécessaires est loin de 10 000 (le nombre total de codes différents composés de 4 chiffres de 0 à 9). Une fois qu'ils ont compris le fonctionnement du voyant lumineux, il est possible de leur demander de réfléchir à la méthode la plus efficace possible. L'algorithme suivant apparaît rapidement : pour rechercher le premier chiffre du code, il suffit de tous les essayer un par un, jusqu'à ce que le voyant s'allume en vert et ainsi être sûr de l'avoir trouvé. Cela prend au pire 10 essais. Connaissant le premier chiffre, il est alors possible de rechercher le deuxième en tapant deux chiffres à la fois jusqu'à ce que le voyant s'allume en vert de nouveau. En suivant cette stratégie, il suffit de 10 autres essais au maximum. Ainsi, au total, en utilisant les informations issues du voyant du digicode, il faut faire $10+10+10+10 = 40$ essais dans le pire des cas pour ouvrir la porte.
Les élèves arrivent le plus souvent à trouver cette stratégie et découvrent ainsi le principe des attaques par canal auxiliaire. En effet, les informations apportées par les couleurs du voyant laissent fuiter des informations permettant de retrouver en seulement quelques essais le code secret, au lieu des 10 000 essais nécessaires dans le pire des cas en utilisant une méthode de recherche exhaustive (« brute force » en anglais).

Il est possible de faire encore un peu mieux ! Pour cela, il suffit de s'apercevoir que 9 essais au maximum permettent de connaître le premier chiffre. En effet, si le voyant s'allume encore en rouge au neuvième essai, alors il ne reste plus qu'une solution possible et il est inutile de faire le dernier essai. Le même raisonnement s'applique pour le deuxième et le troisième chiffres. En revanche, pour le dernier chiffre, si 9 essais au maximum suffisent toujours pour le découvrir, il faut bien ouvrir la porte, et donc taper le code une fois qu'il est découvert. Finalement, $9 + 9 + 9 + 9 + 1 = 37$ essais au maximum permettent d'ouvrir la porte.

La conclusion des élèves après cette activité est souvent que les « vrais » digicodes ne fonctionnent heureusement pas comme celui qui leur est proposé. Cependant, il est important de souligner que des attaques du même type existent réellement sur des systèmes informatiques.

3 Pour aller plus loin : une attaque de RSA par la consommation électrique

Dans cette partie, nous présentons le principe de l'attaque présentée par Paul Kocher dans son article de 1996.

3.1 Le chiffrement RSA

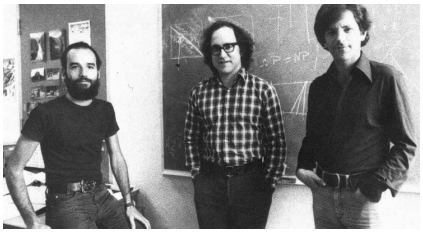

FIGURE 3 – Rivest, Shamir et Adleman en 1978.

En 1978, Ronald Rivest, Adi Shamir et Leonard Adleman (dont les initiales forment l'acronyme RSA) ont inventé le chiffrement RSA, le premier chiffrement à clé publique (ou chiffrement asymétrique), proposé dans l'article [2]. Cette découverte fut une révolution, qui leur a naturellement valu d'obtenir le Prix Turing en 2002. En effet, jusqu'à leur invention, le seul moyen de communiquer de manière chiffrée et donc sécurisée, c'était d'utiliser un chiffrement symétrique. Cela impliquait de connaître son interlocuteur et de s'être mis d'accord au préalable sur une clé secrète partagée. Avec le chiffrement RSA, chaque utilisateur peut diffuser une clé publique de chiffrement accessible à tous. Tout un chacun peut l'utiliser pour chiffrer des messages. En revanche, seul le possesseur de la clé privée de déchiffrement correspondante est capable de déchiffrer ces messages chiffrés. RSA est l'un des chiffrements à clé publique les plus utilisés, mais aussi l'un des plus attaqués.

Les calculs de RSA. Pour générer une paire de clés RSA, il faut d'abord choisir p et q, deux nombres premiers distincts et calculer leur produit $n = p \times q$, appelé *module de chiffrement*. Ensuite, il faut calculer $\Phi(n) = (p-1) \times (q-1)$ qui est appellé l'*indicatrice d'Euler* de n, une notion inventée par le mathématicien suisse Leonhard Euler (1707-1783). Il reste alors à choisir un entier naturel e premier[1] avec $\Phi(n)$ et strictement inférieur à $\Phi(n)$. La clé publique RSA est composée de n et de e. La clé secrète associée est l'entier naturel d, inverse[2] de $e \mod \Phi(n)$, lui aussi strictement inférieur à $\Phi(n)$. Le calcul de d est réalisable efficacement en utilisant l'algorithme d'Euclide étendu.

Le message en clair M doit être strictement inférieur au module n. En effet, les calculs de chiffrement et de déchiffrement s'effectuant $\mod n$, ils donnent des résultats dans $\mathbb{Z}/n\mathbb{Z}$: le message déchiffré est obligatoirement un entier strictement inférieur au module n. Si ce n'est pas le cas du message clair initial M, ce dernier ne pourra pas être retrouvé intact.
Le chiffrement de M avec la clé publique (n, e) s'effectue en calculant :

$$C = M^e \mod n$$

Ensuite, pour déchiffrer C avec la clé secrète d, il faut effectuer le calcul :

$$M = C^d \mod n$$

La preuve que le message initial M est bien le résultat de l'opération de déchiffrement repose sur une propriété d'arithmétique modulaire qui remonte au dix-septième siècle, appelée le petit théorème de Fermat et démontrée par Pierre de Fermat (1607-1685). Les opérations de chiffrement et de

1. Deux entiers sont *premiers entre eux* lorsque leur unique diviseur commun est 1. Par exemple 15 et 28 sont premiers entre eux, mais 15 et 21 ne le sont pas.

2. Deux entiers a et b sont *inverses l'un de l'autre* modulo n lorsque leur produit est égal à 1 modulo n, c'est-à-dire $a \times b = 1 \mod n$. Par exemple, 3 et 9 sont inverses l'un de l'autre modulo 26, car $3 \times 9 = 27 = 1 \mod 26$. Le nombre a possède un inverse modulo n si et seulement si a et n sont premiers entre eux.

déchiffrement de RSA utilisent l'élévation à une puissance, et les nombres impliqués sont très grands, il est donc important que cette opération soit rapide. C'est ce qui explique l'utilisation de l'algorithme d'exponentielle rapide, sur lequel repose l'attaque de Kocher.

La sécurité de RSA. Comme la clé publique est constituée de e et n, il est possible de découvrir les nombres premiers p et q en factorisant le module $n = p \times q$. À partir de là, les calculs de $\Phi(n) = (p-1)(q-1)$, puis de la clé secrète $d = e^{-1} \mod n$ sont rapides. Ainsi, la sécurité du chiffrement RSA repose sur le fait qu'il n'existe pas encore d'algorithme permettant de factoriser efficacement des produits de grands nombres premiers.
Avec l'augmentation de la rapidité des processeurs au cours du temps, des nombres premiers de plus en plus grands sont nécessaires pour assurer la sécurité du chiffrement RSA. Ainsi, la longueur des clés RSA préconisée en 2020 par le NIST et l'ANSSI est au minimum de 2 048 bits et une taille de 4 096 bits est fortement recommandée.
Il faut enfin savoir que l'arrivée de l'ordinateur quantique, si elle se concrétise, mettra fin à la sécurité du chiffrement RSA. En effet, l'algorithme de Shor, inventé en 1994 par Peter W. Shor (1959-), bien avant la réalisation du premier ordinateur quantique, permet de factoriser très rapidement le produit de deux nombres premiers, même très grands. Autrement dit, cet algorithme quantique rend possible de retrouver la clé secrète du chiffrement RSA à partir de la clé publique. Il permet aussi de casser les chiffrements basés sur le logarithme discret, comme le chiffrement d'El Gamal.

3.2 L'algorithme d'exponentielle rapide

Rappelons qu'Alice possède une clé de chiffrement RSA publique, composée de deux entiers e et n, et une clé de déchiffrement RSA privée, composée du seul nombre d. Bien entendu, les nombres e, n et d ne sont pas choisis n'importe comment, comme expliqué précédemment. De plus, pour assurer la sécurité de ce chiffrement, ces nombres doivent être très grands (plusieurs centaines de chiffres).
Lorsqu'Alice reçoit un message secret C qui a été chiffré avec sa clé de chiffrement RSA publique (e et n), elle (ou plutôt le processeur de son ordinateur) le déchiffre en calculant le reste de la division de C^d par n. Le message clair obtenu est un entier compris entre 0 et $n-1$.

Comme d est très grand, ce calcul ne peut pas se faire simplement en multipliant $C \times C \times C \times \ldots \times C \times C$ (d fois), puis en faisant la division du résultat par n, cela prendrait trop de temps de faire les $d-1$ multiplications nécessaires. Heureusement, il existe des algorithmes qui permettent d'aller plus vite. L'un d'eux s'appelle l'*algorithme d'exponentielle rapide*. L'exemple ci-dessous décrit son fonctionnement.

Pour calculer $5^{13} \mod 187$, il faut commencer par décomposer l'exposant 13 en une somme de puissances de 2, ce qui donne $13 = 8 + 4 + 1 = 2^3 + 2^2 + 2^0$. Cette décomposition correspond à l'écriture de 13 en binaire : 1101.
Ceci permet d'écrire :

$$5^{13} = 5^{(2^3+2^2+2^0)} = 5^{(2^3)} \times 5^{(2^2)} \times 5^{(2^0)}$$

Deux propriétés des puissances sont maintenant nécessaires :

- $a^{\left(2^0\right)} = a^1 = a$
- $a^{\left(2^{(b+1)}\right)} = a^{\left(\left(2^b\right)\times 2\right)} = \left(a^{\left(2^b\right)}\right)^2$

pour n'importe quel entier a.

L'algorithme d'exponentielle rapide consiste à calculer les $5^{\left(2^i\right)}$ successifs par des élévations au carré, tout en faisant au fur et à mesure les multiplications nécessaires pour obtenir 5^{13}.

- au départ, la valeur du produit est 1 ;
- ensuite, $5^{\left(2^0\right)} = 5$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $1\times 5 = 5$;
- calcul de $5^{\left(2^1\right)} = \left(5^{\left(2^0\right)}\right)^2 = 5^2 = 25$; il n'est pas utilisé dans le produit ;
- calcul de $5^{\left(2^2\right)} = \left(5^{\left(2^1\right)}\right)^2 = 25^2 = 625$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $5 \times 625 = 3\,125$;
- calcul de $5^{\left(2^3\right)} = \left(5^{\left(2^2\right)}\right)^2 = 625^2 = 390\,625$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $3\,125 \times 390\,625 = 1\,220\,703\,125$, et ceci termine le caclul.

Les opérations effectuées pour calculer $5^{13} = 1\,220\,703\,125$ avec cet algorithme sont donc dans l'ordre : un produit, deux élévations au carré, un produit, une élévation au carré et un dernier produit.

Comme ce qui nous intéresse, ce n'est pas exactement la valeur de 5^{13}, mais plutôt le reste de la division de 5^{13} par 187, en cours de calcul, tous les nombres (aussi bien les carrés que les valeurs intermédiaires du produit) qui atteignent ou dépassent 187 peuvent être remplacés par le reste de leur division par 187. Le calcul devient alors :

- au départ, la valeur du produit est $1 < 187$;
- ensuite, $5^{\left(2^0\right)} = 5 < 187$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $1 \times 5 = 5 < 187$;
- calcul de $5^{\left(2^1\right)} = \left(5^{\left(2^0\right)}\right)^2 = 5^2 = 25 < 187$; il n'est pas utilisé dans le produit ;
- calcul de $5^{\left(2^2\right)} = \left(5^{\left(2^1\right)}\right)^2 = 25^2 = 625 \geq 187$; il est remplacé par $625 \mod 187 \equiv 64$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $5 \times 64 = 320 \geq 187$; il est remplacé par $320 \mod 187 \equiv 133$
- calcul de $5^{\left(2^3\right)} = \left(5^{\left(2^2\right)}\right)^2 = 64^2 = 4\,096 \geq 187$; il est remplacé par $4\,096 \mod 187 \equiv 169$; il est utilisé dans le produit, donc la nouvelle valeur du produit est $133\times 169 = 22\,477 \geq 187$; il est remplacé par $22\,477 \mod 187 \equiv 37 < 187$, et ceci termine le calcul.

Bien sûr, cet algorithme est relativement compliqué, mais il a permis de faire seulement 3 élevations au carré et 3 multiplications pour calculer 5^{13} au lieu des 12 multiplications nécessaires en calculant plus simplement $5 \times 5 \times 5 \times ... \times 5$. Il faut savoir également que cet algorithme est très efficace sur des grands nombres, car plus l'exposant est grand, plus le gain de nombre d'opérations à effectuer augmente.

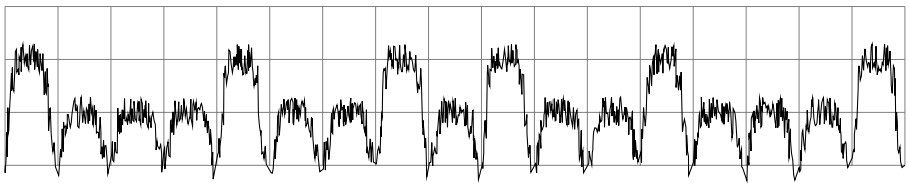

FIGURE 4 – Consommation électrique du processeur d'Alice.

3.3 La consommation électrique laisse fuir des informations

L'attaque proposée par Paul Kocher exploite le fait que, dans un processeur, le circuit électronique calculant un carré est différent de celui calculant un produit de deux nombres différents, et le second calcul consomme plus d'énergie que le premier.
La figure 4 représente la trace de la consommation électrique du processeur de l'ordinateur d'Alice pendant le déchiffrement d'un message chiffré par la méthode RSA avec l'algorithme d'exponentielle rapide. Elle permet de découvrir la clé privée d'Alice.

En effet, nous venons de voir qu'un processeur consomme moins d'énergie pour calculer un carré que pour calculer un produit. Par conséquent, sur le tracé proposé, un petit pic correspond à un carré et un grand pic correspond à un produit. En utilisant cette correspondance, les opérations successives effectuées par le processeur pendant le calcul de C^d sont : produit ; carré ; carré ; carré ; produit ; carré ; carré ; produit ; carré ; produit ; carré ; carré ; produit ; carré ; carré ; carré ; produit.

Ces informations ne donnent aucun renseignement sur C, mais en suivant la description de l'algorithme d'exponentielle rapide, elles permettent de découvrir quelles sont les puissances de 2 qui interviennent dans la décomposition de l'exposant d : ce sont celles qui se trouvent juste avant un produit (sauf pour la première, puisque $C^{(2^0)} = C$ n'est pas un carré).

- produit : 2^0 intervient dans d
- carré : calcul de $C^{(2^1)}$
- carré : calcul de $C^{(2^2)}$
- carré : calcul de $C^{(2^3)}$
- produit : 2^3 intervient dans d
- carré : calcul de $C^{(2^4)}$
- carré : calcul de $C^{(2^5)}$
- produit : 2^5 intervient dans d
- carré : calcul de $C^{(2^6)}$
- produit : 2^6 intervient dans d
- carré : calcul de $C^{(2^7)}$
- carré : calcul de $C^{(2^8)}$
- produit : 2^8 intervient dans d
- carré : calcul de $C^{(2^9)}$
- carré : calcul de $C^{(2^{10})}$
- carré : calcul de $C^{(2^{11})}$

- produit : 2^{11} intervient dans d

Finalement, la clé de déchiffrement d'Alice est

$$\begin{aligned} d &= 2^0 + 2^3 + 2^5 + 2^6 + 2^8 + 2^{11} \\ &= 1 + 8 + 32 + 64 + 256 + 2048 \\ &= 2409 \end{aligned}$$

3.4 Une course aux armements

L'attaque de RSA présentée ci-dessus est une attaque *par analyse simple de la consommation*. Il est possible de s'en prémunir en modifiant l'algorithme de l'exponentielle rapide pour qu'il effectue à chaque étape à la fois une élévation au carré et une multiplication. Ainsi, l'attaquant ne peut plus déterminer quelles sont les puissances de 2 qui interviennent dans la clé de déchiffrement d en analysant la consommation électrique du processeur.
Cela peut être fait simplement en effectuant à part, en guise de leurre, le produit de tous les $C^{(2^i)}$ qui ne sont pas utiles pour le calcul de C^d. Malheureusement, dans ce cas, le calcul est vulnérable à une attaque dite par *injection de fautes*, qui consiste à perturber physiquement l'exécution de l'algorithme au moment d'une multiplication. Si la perturbation modifie le résultat final, c'est que la multiplication concernée est vraiment nécessaire au calcul de C^d, dans le cas contraire, c'est qu'il s'agit d'une multiplication leurre. Il existe une autre variante plus sophistiquée de l'algorithme d'exponentielle rapide, appelée l'*échelle de Montgomery*, dont l'exécution effectue à chaque étape à la fois une élévation au carré et une multiplication, sans qu'aucune des multiplications ne soit inutile. Malheureusement, ce dernier algorithme est encore vulnérable à une attaque dite par *analyse différentielle de consommation*.

4 Conclusion

La problématique des attaques par canal auxiliaire est prise très au sérieux par les spécialistes de sécurité informatique. Ainsi, dans l'article [3], publié en 2023, les auteurs testent 34 outils d'analyse de code qui annoncent détecter des vulnérabilités à certaines attaques de ce type. Ils proposent ensuite un ensemble de recommandations à destination de la communauté cryptographique pour améliorer l'efficacité de ces outils. Également en 2023, la chercheuse Sandrine Blazy a reçu la médaille d'argent du CNRS pour ses travaux sur le développement de compilateurs garantissant la génération d'exécutables qui n'ajoutent pas de vulnérabilités aux programmes.

Adaptation

- Pour des lycéens ou des étudiants qui connaissent le chiffrement RSA et l'algorithme d'exponentielle rapide, il est possible de présenter l'activité sur l'attaque de RSA par la consommation électrique.

Références

[1] Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, Paul C. Kocher, Advances in Cryptology - CRYPTO'96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings, Neal Koblitz, Lecture Notes in Computer Science, Vol. 1109, pages 104–113, Springer, 1996.

[2] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, Commun. ACM, Vol. 21-2, pages 120-126, Association for Computing Machinery, New York, NY, USA, 1978.

[3] A Systematic Evaluation of Automated Tools for Side-Channel Vulnerabilities Detection in Cryptographic Libraries, Antoine Geimer, Mathéo Vergnolle, Frédéric Recoules, Lesly-Ann Daniel, Sébastien Bardin, Clémentine Maurice, Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, pages 1690–1704, 2023.