INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Enseignant : *Vincent Migliore*
vincent.migliore@insa-toulouse.fr

Securité Matérielle - SCA2 - Lien consommation - algorithmique - architecture

1 Introduction

Durant les expérimentations relatives à l'attaque SPA sur digicode, vous avez pu voir que la consommation énergétique permettait de déduire des informations sur des variables manipulées au niveau algorithmique. En particulier, dans le cas du digicode, nous avions pu corréler la trace de consommation avec la durée d'exécution de la boucle principale de la fonction de vérification du code, permettant de savoir si certains chiffres du code étaient corrects.

Une solution permettant de limiter la portée de l'attaque avait été proposé, consistant à rendre le temps d'exécution constant, ce qui effectivement permettait d'éviter cette attaque. Nous allons montrer dans ce TP que ce n'est malheureusement pas suffisant.

2 Définition de la cible

La cible est identique au TP sur la SPA. Le banc d'expérimentation est le Chipwhisperer, ciblant un micro-contrôleur Cortex-M (vous pouvez vous référer au TP précédent pour la mise en place du banc).

3 Organisation des expérimentations

Le TP est découpé en un certain nombre de **labs** à faire dans un ordre précis. La structure des ressources du TP est la suivante :

lab*i* : Dossier contenant les ressources nécessaires à chaque lab. Il est en général composé de fichiers sources pour compiler le firmware du cortex-M de l'expérimentation, et d'un fichier python **test.py** permettant de lancer l'expérimentation.

Makefile : Fichier permettant de compiler les firmwares de chaque lab. Son utilisation est la suivante : **make** LAB=*nom-du-dossier-du-lab*

Exemple : make LAB=lab1

setup.sh : Script à lancer dès que possible permettant de télécharger le compilateur gcc pour ARM et l'installer dans le dossier courant.

lab 1

Ce premier lab a pour objectif de reprendre en main les outils du Chipwhisperer et d'étudier la consommation d'un code élémentaire.

Ouvrez le fichier **lab1/main.c** et observez le contenu de la fonction **test()**. Faite une hypothèse sur la forme que devrait prendre la consommation énergétique à partir de la séquence d'opérations. Compilez le firmware et lancez l'expérimentation. Analysez le résultat. Que constatez-vous et qu'en pensez-vous ?

lab 2

Dans ce deuxième lab, nous allons voir l'impact d'un code similaire au **lab1** mais avec une structure algorithmique différente.

Complétez le fichier **lab2/main.c** pour faire la même opération mais cette fois-ci avec une boucle **for**. On fera bien attention à ajouter à chaque variable le mot-clé **volatile** pour éviter toute optimisation par le compilateur. Compilez le firmware et lancez l'expérimentation. Analysez le résultat. Que constatez-vous et qu'en pensez-vous, surtout vis-à-vis du **lab1** ?

lab 3

Dans ce troisième lab, nous allons étudier s'il existe une corrélation entre la consommation énergétique et les variables manipulées, même dans le cas d'une implémentation en temps constant. Pour cela, nous allons suivre une méthodologie qui est proche de la méthode dite du *fixed vs random test vector leakage assessment*. L'objectif est de comparer 2 familles de traces, une première famille composée de traces paramétrées par une entrée qui est fixe, et une seconde famille composée de traces paramétrées par une entrée qui est aléatoire.

Ouvrez le fichier **lab3/main.c**. Est-ce que l'implémentation sera en temps constant ? Comment le code proposé peut servir à identifier s'il existe une corrélation entre la consommation énergétique et les variables manipulées ?

Complétez le fichier **lab3/test.py** pour appliquer la méthodologie de test expliquée dans le premier paragraphe, lancez l'expérimentation et analysez le résultat.

lab 4

Ce lab est dédié à l'étude plus théorique de l'identification des fuites dans un micro-contrôleur. Une partie importante de la consommation est due au changement d'état des transistors. Les registres étant composés de plusieurs transistors synchronisés sur la même horloge, le changement de la valeur des bits du registre provoque également une surconsommation. A partir du schéma ci-dessous et de vos connaissances, étudiez les différents cas de figure qui peuvent provoquer des fuites de donnée via l'écoute de la consommation.