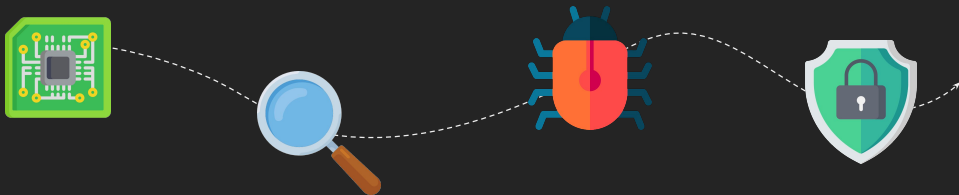


Cybersécurité à la loupe : du logiciel au matériel



Clémentine Maurice, chargée de recherche CNRS
Laboratoire CRISTAL, équipe-projet Inria Spirals
01/10/2025 - Bar des Sciences, Université de Lille

PARCOURS

2007-2012

Double diplôme **ingénieur** informatique
et **master recherche**, INSA Rennes

2012-2015

Thèse CIFRE, Technicolor/EURECOM

2016-2017

Postdoctorat, Graz University of Technology,
Autriche

depuis 2017

Chercheuse au CNRS

2017-2021 : à l'IRISA, Rennes

depuis 2021 : à CRIStAL, Lille

La ville de Lille victime d'une cyberattaque, quatre agents municipaux ont reçu une demande de rançon

L'attaque informatique, qui empêche notamment l'accès au standard de la municipalité, est en cours depuis le 28 février.



Publié le 15/03/2023 22:16

⌚ Temps de lecture : 1 min

Cyberattaque à l'hôpital de Cannes : cartes d'identité, bilans médicaux, bulletins de salaire... 61 gigaoctets de données publiés sur internet

L'activité de l'hôpital est presque revenue à la normale et le système d'information est en train d'être rétabli.



Publié le 02/05/2024 14:55

⌚ Temps de lecture : 1 min

CYBERSÉCURITÉ \ FORMATION \ DONNÉES PERSONNELLES

Cybersécurité : l'université Paris-Saclay touchée par un ransomware

L'université francilienne a annoncé avoir été visée par une attaque par ransomware le 11 août. Un incident qui intervient une semaine après une autre cyberattaque contre plus de 40 établissements culturels en France.

Yoann Bourgin

13 août 2024 - 10h30



Paris 2024 : 548 signalements et incidents de cybersécurité liés aux Jeux décomptés entre mai et septembre

D'après l'Agence nationale de la sécurité des systèmes d'information, ces alertes de cybersécurité n'ont eu que de "faibles impacts".



Publié le 10/09/2024 18:20

⌚ Temps de lecture : 1 min

Cyberattaque à l'hôpital d'Armentières : 300 000 patients concernés par le vol de données

L'établissement avait dû fermer temporairement ses urgences à la suite de cette cyberattaque le 11 février.



Publié le 28/02/2024 18:51 | Mis à jour le 28/02/2024 19:14

⌚ Temps de lecture : 1 min

Cultura victime d'une cyberattaque, les données de 1,5 million de clients dérobées

La fuite de données concerne le nom, prénom, numéro de téléphone, adresse e-mail et postale ainsi que le contenu des commandes des clients. Les mots de passe et données bancaires n'ont en revanche pas été compromis, selon l'enseigne.

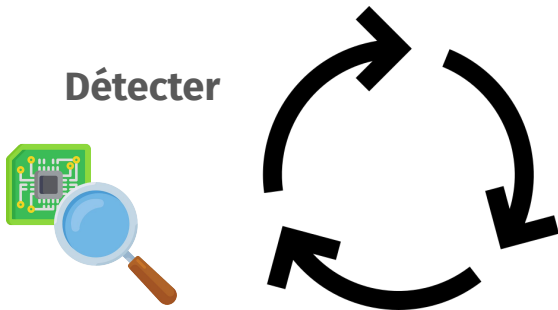


Publié le 10/09/2024 20:39

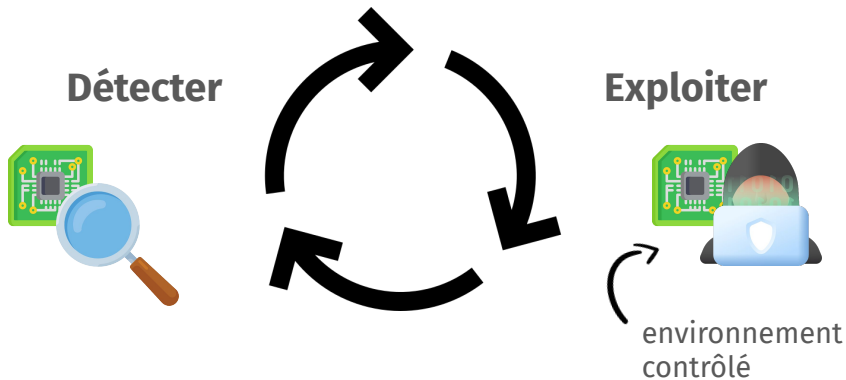
⌚ Temps de lecture : 1 min

La sécurité n'est pas qu'une affaire de logiciels !

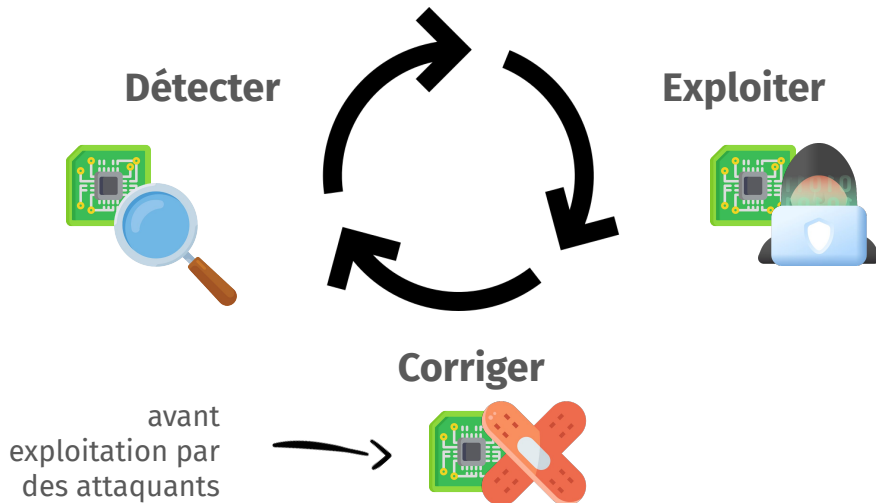
Ma recherche : sécurité à l'interface entre le logiciel et le matériel



Ma recherche : sécurité à l'interface entre le logiciel et le matériel



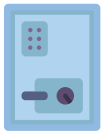
Ma recherche : sécurité à l'interface entre le logiciel et le matériel



Attaques matérielles



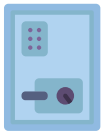
Attaques matérielles



attaques actives: destruction du coffre-fort

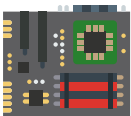
attaques passives: écoute des mécanismes internes

Attaques matérielles



attaques actives: destruction du coffre-fort

attaques passives: écoute des mécanismes internes



attaques actives: laser, perturbation d'horloge...

attaques passives: timing, consommation de courant...

- matériel habituellement modélisé comme couche abstraite correcte

Attaques matérielles

- matériel habituellement modélisé comme couche abstraite correcte, mais attaques possibles

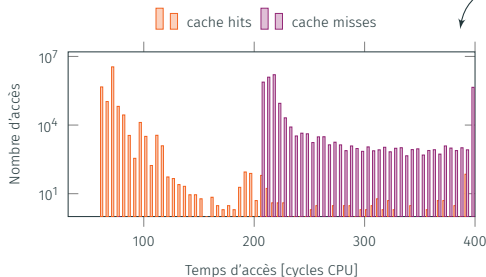
Attaques matérielles

- matériel habituellement modélisé comme couche abstraite correcte, mais attaques possibles
 - par faute : contourner les protections logicielles par des erreurs sur le matériel
 - par canal auxiliaire : observer les effets de bord du matériel sur les calculs

Attaques matérielles

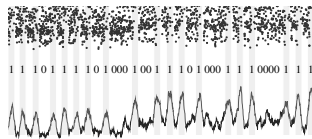
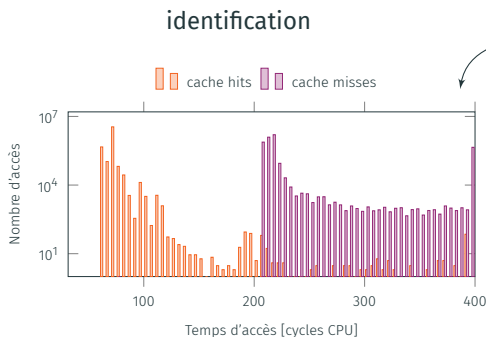
- **matériel** habituellement modélisé comme couche abstraite correcte, mais attaques possibles
 - par faute : contourner les protections logicielles par des **erreurs sur le matériel**
 - par canal auxiliaire : observer les **effets de bord** du matériel sur les calculs

identification



Attaques matérielles

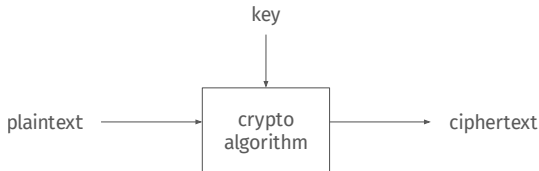
- **matériel** habituellement modélisé comme couche abstraite correcte, mais attaques possibles
 - par faute : contourner les protections logicielles par des **erreurs sur le matériel**
 - par canal auxiliaire : observer les **effets de bord** du matériel sur les calculs



- récupérer clés secrètes, intervalles de frappes de clavier
- contourner sécurité système (ASLR)

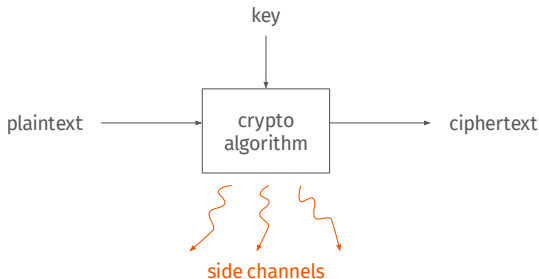
Attaques par canaux auxiliaires

- exploitent l'implémentation d'un système
- basés sur des canaux qui sont en dehors de la spécification fonctionnelle, i.e., qui ne sont pas censés véhiculer d'information utiles
- mais ces canaux peuvent faire fuiter des informations secrètes



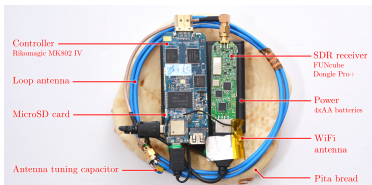
Attaques par canaux auxiliaires

- exploitent l'implémentation d'un système
- basés sur des canaux qui sont en dehors de la spécification fonctionnelle, i.e., qui ne sont pas censés véhiculer d'information utiles
- mais ces canaux peuvent faire fuiter des informations secrètes



Attaques physiques vs attaques microarchitecturales

Attaques par le matériel a.k.a attaques physiques



Accès physique au matériel
→ appareils embarqués

VS

Attaques par le logiciel a.k.a attaques microarchitecturales



Attaque à distance
→ systèmes complexes

De petites optimisations aux attaques par canaux auxiliaires...



- nouvelle microarchitecture tous les ans

De petites optimisations aux attaques par canaux auxiliaires...



- nouvelle microarchitecture tous les ans
- amélioration de performance $\approx 5\%$

De petites optimisations aux attaques par canaux auxiliaires...



- nouvelle microarchitecture tous les ans
- amélioration de performance $\approx 5\%$
- très **petites optimisations**: caches, prédicteurs de branchements...

De petites optimisations aux attaques par canaux auxiliaires...



- nouvelle microarchitecture tous les ans
- amélioration de performance $\approx 5\%$
- très **petites optimisations**: caches, prédicteurs de branchements...
- les attaques viennent de ces optimisations

De petites optimisations aux attaques par canaux auxiliaires...



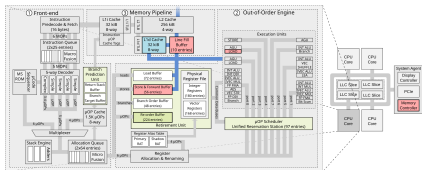
- nouvelle microarchitecture tous les ans
- amélioration de performance $\approx 5\%$
- très **petites optimisations**: caches, prédicteurs de branchements...
- les attaques viennent de ces optimisations
- un attaquant obtient des informations d'un processus victime (vulnérable) en **espionnant l'usage du matériel**

Attaques par canaux auxiliaires : les deux faces d'une même pièce

Matériel



Logiciel



&

Algorithm 1: Square-and-multiply exponentiation

Input: base b , exponent e , modulus n

Output: $b^e \bmod n$

$X \leftarrow 1$

for $i \leftarrow \text{bitlen}(e)$ downto 0 do

$X \leftarrow \text{multiply}(X, X)$

 if $e_i = 1$ then

$X \leftarrow \text{multiply}(X, b)$

 end

end

return X

RQ1. Quels **composants matériels** sont vulnérables...

... et comment les exploiter pour faire fuir des données ?

RQ2. Quelles **implémentations logicielles** sont vulnérables...

... et comment mener ces attaques en pratique ?

RQ1 : Un exemple d'attaque

Algorithm 1: GnuPG 14.13 Square-and-multiply exponentiation

Input: base c , exponent d , modulus n

Output: $c^d \bmod n$

$X \leftarrow 1$

for $i \leftarrow \text{bitlen}(d)$ downto 0 do

$X \leftarrow \text{square}(X)$

$X \leftarrow X \bmod n$

 if $d_i = 1$ then

$X \leftarrow \text{multiply}(X, c)$

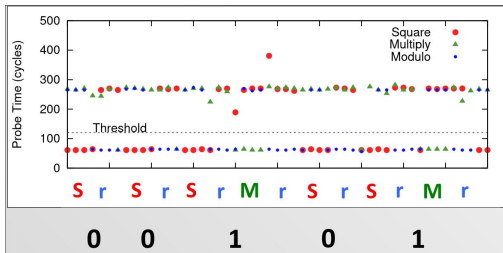
$X \leftarrow X \bmod n$

 end

end

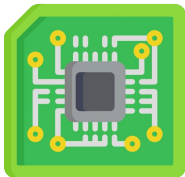
return X

valeur secrète !

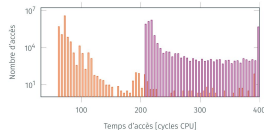


Qu'est-ce qui vient de se passer ?

attaque sur le cache

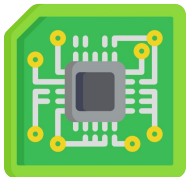


exploite les différences
de temps des accès à la
mémoire

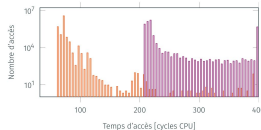


Qu'est-ce qui vient de se passer ?

attaque sur le cache



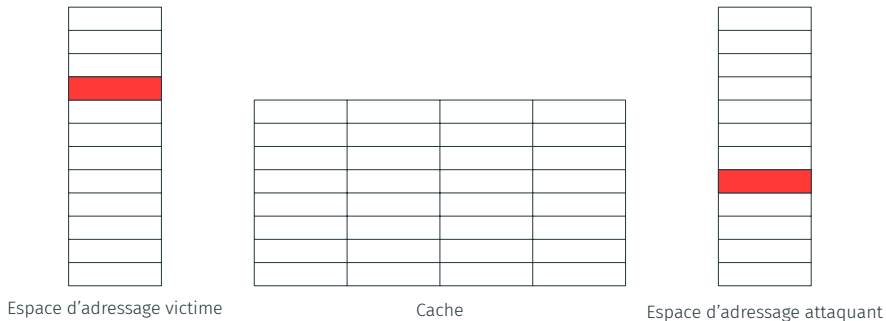
exploite les différences
de temps des accès à la
mémoire



attaquant surveille les
lignes accédées par la
victime, pas le contenu

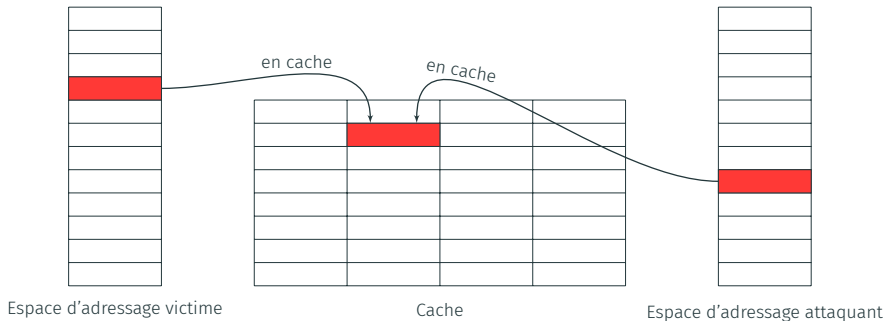


L'attaque sur le cache Flush+Reload



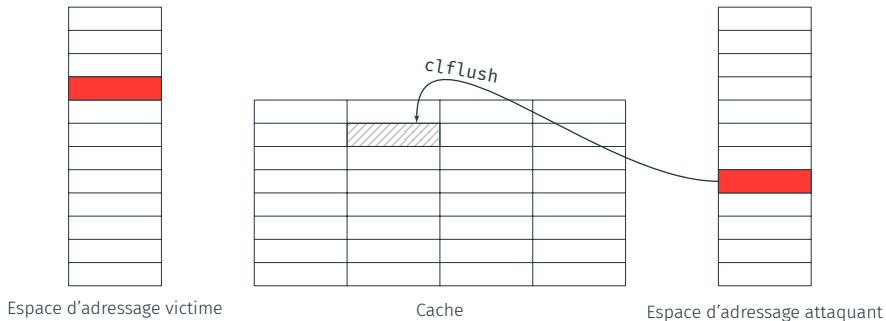
#1 : Mémoire partagée entre l'attaquant et la victime → cache partagé

L'attaque sur le cache Flush+Reload



#1 : Mémoire partagée entre l'attaquant et la victime → cache partagé

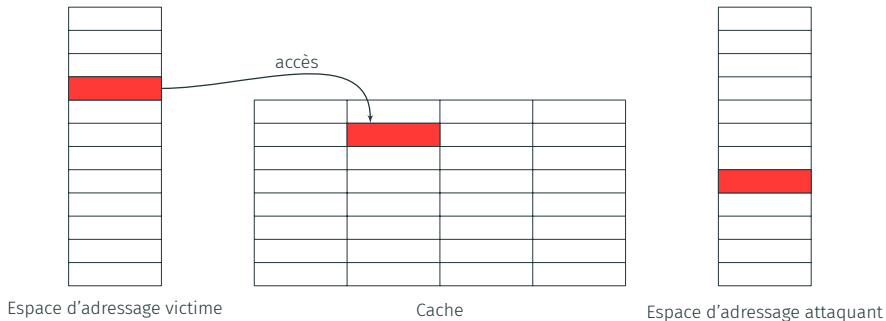
L'attaque sur le cache Flush+Reload



#1 : Mémoire partagée entre l'attaquant et la victime → cache partagé

#2 : Attaquant **flush**e la ligne de cache partagée

L'attaque sur le cache Flush+Reload

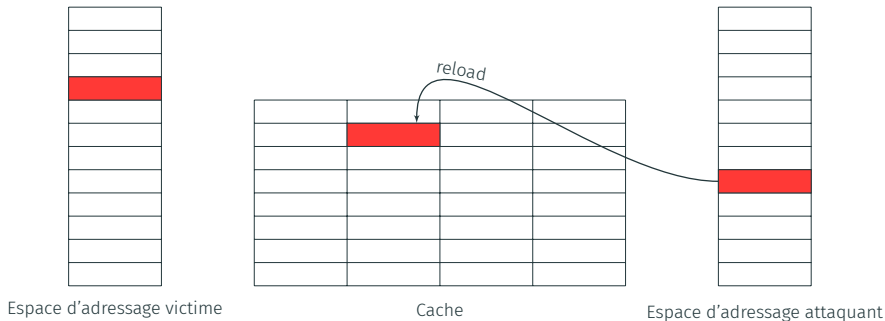


#1 : Mémoire partagée entre l'attaquant et la victime → cache partagé

#2 : Attaquant **flush** la ligne de cache partagée

#3 : Victime accède à une donnée

L'attaque sur le cache Flush+Reload



#1 : Mémoire partagée entre l'attaquant et la victime → cache partagé

#2 : Attaquant **flush**e la ligne de cache partagée

#3 : Victime accède à une donnée

#4 : Attaquant **recharge (reload)** la ligne : apprend l'accès de la victime

Flush+Reload en pratique ?

```
1  int probe(char *adrs) {
2      volatile unsigned long time;
3
4      asm __volatile__ (
5          " mfence                \n"
6          " lfence                \n"
7          " rdtsc                 \n"
8          " lfence                \n"
9          " movl %%eax, %%esi      \n"
10         " movl (%1), %%eax       \n"
11         " lfence                \n"
12         " rdtsc                 \n"
13         " subl %%esi, %%eax       \n"
14         " clflush 0(%1)         \n"
15         : "=a" (time)
16         : "c" (adrs)
17         : "%esi", "%edx");
18     return time < threshold;
19 }
```

Flush+Reload en pratique ?

```
1  int probe(char *adrs) {
2      volatile unsigned long time;
3
4      asm __volatile__ (
5          " mfence                \n"
6          " lfence                \n"
7          " rdtsc                 \n"
8          " lfence                \n"
9          " movl %%eax, %%esi      \n"
10         " movl (%1), %%eax       \n"
11         " lfence                \n"
12         " rdtsc                 \n"
13         " subl %%esi, %%eax      \n"
14         " clflush 0(%1)         \n"
15         : "=a" (time)
16         : "c" (adrs)
17         : "%esi", "%edx");
18     return time < threshold;
19 }
```

horloge

accès mémoire

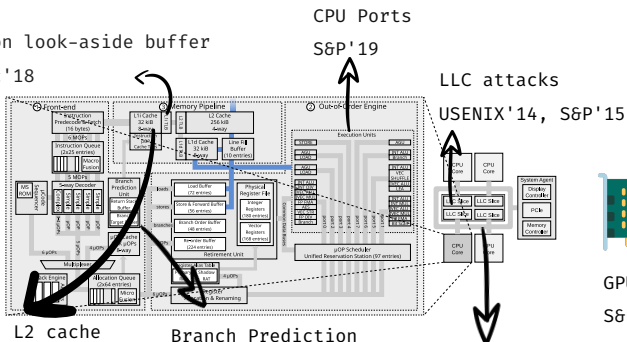
horloge

flush cache

Conclusion RQ1

Translation look-aside buffer

USENIX Sec'18



L1d, L1i, L2 cache

BSDCon'05, CT-RSA'06,

ASIACCS'20

Branch Prediction

CT-RSA'07

CPU Ports

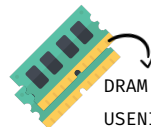
S&P'19

LLC attacks

USENIX'14, S&P'15

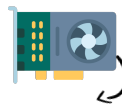
Ring Interconnect

USENIX Sec'21, DIMVA'21



DRAM

USENIX Sec'16



GPU

S&P'18

État de l'art aujourd'hui: chaque composant partagé par deux processus est un potentiel vecteur d'attaque par canal auxiliaire

Problème ?

Vulnérabilité canal auxiliaire

Tout **branchement** ou **accès**
mémoire dépendant d'un **secret**

RQ2 : Comment on fixe ça dans les logiciels ?

Problème ?

Vulnérabilité canal auxiliaire

Tout **branchement** ou **accès**
mémoire dépendant d'un **secret**



Solution !

Programmation *constant time*

Pas de **branchement** ou d'**accès**
mémoire dépendant d'un **secret**

RQ2 : Comment on fixe ça dans les logiciels ?

Problème ?

Vulnérabilité canal auxiliaire

Tout **branchement** ou **accès**
mémoire dépendant d'un **secret**



Solution !

Programmation *constant time*

Pas de **branchement** ou d'**accès**
mémoire dépendant d'un **secret**

Ça a l'air facile non ?

LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage

Diego F. Aranha
DIGIT, Aarhus University
Denmark
dfaranha@eng.au.dk

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Akira Takahashi
DIGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Mehdi Tibouchi
NTT Corporation
Japan
mehdi.tibouchi.br@hco.ntt.co.jp

Yuval Yarom
University of Adelaide and Data61
Australia
yval@cs.adelaide.edu.au

ABSTRACT

Although it is one of the most popular signature schemes today, ECDSA presents a number of implementation pitfalls, in particular due to the very sensitive nature of the random value (known as the *nonce*) generated as part of the signing algorithm. It is known that any small amount of nonce exposure or nonce bias can in principle lead to a full key recovery: the key recovery is then a particular instance of Boneh and Venkatesan's *hidden number problem* (HNP). That observation has been practically exploited in many attacks in the literature, taking advantage of implementation defects or side-channel vulnerabilities in various concrete ECDSA implementations. However, most of the attacks so far have relied on at least 2

ephemeral random value called *nonce*, which is particularly sensitive: it is crucial to make sure that the nonces are kept in secret and sampled from the uniform distribution over a certain integer interval. It is easy to see that if the nonce is exposed or reused completely, then an attacker is able to extract the secret signing key by observing only a few signatures. By extending this simple observation, cryptanalysts have discovered stronger attacks that make it possible to recover the secret key even if short bit substrings of the nonces are leaked or biased. These extended attacks relate key recovery to the so-called hidden number problem (HNP) of Boneh and Venkatesan [15], and are part of a line of research initiated by Howgrave-Graham and Smart [36], who described a lattice-based

LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage

Diego F. Aranha
DIGIT, Aarhus University
Denmark
dfaranha@eng.au.dk

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Akira Takahashi
DIGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Yuval Yarom
University of Adelaide and Data61
Australia
yval@cs.adelaide.edu.au

Mehdi Tibouchi
NTT Corporation
Japan
mt@hco.ntt.co.jp

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

Luke Valenta
University of Pennsylvania
lukev@cis.upenn.edu

ABSTRACT

In recent years, applications increasingly adopt security primitives designed with better countermeasures against side channel attacks. A concrete example is Libcrypt's implementation of ECDSA encryption with Curve25519. The implementation employs the Montgomery ladder scalar-by-point multiplication, uses the unified, branchless Montgomery double-and-add formula and implements a constant-time argument swap within the ladder. However, Libcrypt's field arithmetic operations are not implemented in a constant-time side-channel-resistant fashion. Based on the secure design of Curve25519, users of the curve are advised that there is no need to perform validation of input points. In this work we demonstrate that when this recommendation is followed, the mathematical structure of Curve25519 facilitates the exploitation of side-channel weaknesses.

implementations. A particular threat arises from asynchronous attacks, where the attacker only has to execute a program concurrently with the victim's program (on the same physical CPU) in order to collect temporal information about the victim's behavior. With this temporal information at hand, the attacker can recover the internal workings of the victim. Because microarchitectural attacks execute on the same processor as the victim, the attacker can only achieve limited temporal resolution. Typically, the attacker can only distinguish between event timings if the events are several hundreds or thousands of ten target key-dependent variations in either the order of high-level operations or in their arguments. More specifically, such attacks usually target the square-and-multiply sequence of the modular exponentiation in RSA [61, 72], ElGamal [55, 73] and DSA [63], or

ephemeral random value called *nonce*, which is particularly sensitive: it is crucial to make sure that the nonces are kept in secret and sampled from the uniform distribution over a certain integer interval. It is easy to see that if the nonce is exposed or reused completely, then an attacker is able to extract the secret signing key by observing only a few signatures. By extending this simple observation, cryptanalysts have discovered stronger attacks that make it possible to recover the secret key even if short bit substrings of the nonces are leaked or biased. These extended attacks relate key recovery to the so-called hidden number problem (HNP) of Boneh and Venkatesan [15], and are part of a line of research initiated by Howgrave-Graham and Smart [36], who described a lattice-based

LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage

Diego F. Aranha
DiGIT, Aarhus University
Denmark
dfaranha@eng.au.dk

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Akira Takahashi
DiGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Yuval Yarom
University of Adelaide and Data61
Australia
yval@cs.adelaide.edu.au

Mehdi Tibouchi
NTT Corporation
Japan
mtibouchi@ntt.co.jp

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

Luke Valenta
University of Pennsylvania
lukev@cis.upenn.edu

ABSTRACT

In recent years, applications increasingly adopt security primitives designed with better countermeasures against side channel attacks. A concrete example is Libcrypt's implementation of ECDSA encryption with Curve25519. The implementation employs the Montgomery ladder scalar-by-point multiplication, uses the unified, branchless Montgomery double-and-add formula and implements a constant-time argument swap within the ladder. However, Libcrypt's field arithmetic operations are not implemented in a constant-time side-channel-resistant fashion. Based on the secure design of Curve25519, users of the curve are advised that there is no need to perform validation of input points. In this work we demonstrate that when this recommendation is followed, the mathematical structure of Curve25519 facilitates the exploitation of side-channel weaknesses.

implementations. A particular threat arises from asynchronous attacks, where the attacker only has to execute a program concurrently with the victim's program (on the same physical CPU) in order to collect temporal information about the victim's behavior. With this temporal information at hand, the attacker can reconstruct the internal workings of the victim. Because microarchitectural attacks execute on the victim as the victim, the attacker can only observe the execution event timings if the victim's execution cycles are not separated by ten target key-dependent operations or in their usual timing. Typically, the attacker usually targets the squaring exponentiation in RSA [6].

...hemes today, ds, in particular ae (known as the a. It is known that a. can in principle / is then a particular number problem (HNP). exploited in many attacks. implementation defects or concrete ECDSA implementation far have relied on at least 2

ephemeral random value called *nonce*, which is particularly sensitive: it is crucial to make sure that the nonces are kept in secret and sampled from the uniform distribution over a certain integer interval. It is easy to see that if the nonce is exposed or reused completely, then an attacker is able to extract the secret signing key by observing only a few signatures. By extending this simple observation, cryptanalysts have discovered stronger attacks that make it possible to recover the secret key even if short bit substrings of the nonces are leaked or biased. These extended attacks relate key recovery to the so-called hidden number problem (HNP) of Boneh and Venkatesan [15], and are part of a line of research initiated by Howgrave-Graham and Smart [36], who described a lattice-based

PARASITE: Password Recovery Attack against Srp Implementations in The wild

Daniel De Almeida Braga
daniel.de-almeida-braga@irisa.fr
Univ Rennes, CNRS, IRISA
Rennes, France

Pierre-Alain Fouque
pa.fouque@gmail.com
Univ Rennes, CNRS, IRISA
Rennes, France

Mohamed Sabt
mohamed.sabt@irisa.fr
Univ Rennes, CNRS, IRISA
Rennes, France

ABSTRACT

Protocols for password-based authenticated key exchange (PAKE) allow two users sharing only a short, low-entropy password to establish a secure session with a cryptographically strong key. The challenge in designing such protocols is that they must resist offline dictionary attacks in which an attacker exhaustively enumerates

KEYWORDS

SRP; PAKE; Flush+Reload; PDA; OpenSSL; micro-architectural attack

ACM Reference Format:

Daniel De Almeida Braga, Pierre-Alain Fouque, and Mohamed Sabt. 2021.

Des attaques partout...

LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage

Diego F. Aranha
University of Campinas
Denmark
dfaranha@eng.au.dk

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Akira Takahashi
DIGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Mehdi Tibouchi
NTT Corporation
Japan
mehdi.tibouchi@ntt.co.jp

Yuval Yarom
University of Adelaide and Data61
Australia
yval@cs.adelaide.edu.au

May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

Luke Valenta
University of Pennsylvania
lukev@cis.upenn.edu

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

ABSTRACT

In recent years, applications increasingly adopt security primitives designed with better countermeasures against side channel attacks. A concrete example is Libcrypt's implementation of ECDH encryption with Curve25519. The implementation employs the Montgomery ladder scalar-by-point multiplication, uses the unified, branchless Montgomery double-and-add formula and implements a constant-time argument swap within the ladder. However, Libcrypt's field arithmetic operations are not implemented in a constant-time side-channel-resistant fashion. Based on the secure design of Curve25519, users of the curve are advised that there is no need to perform validation of input points. In this work we demonstrate that when this recommendation is followed, the mathematical structure of Curve25519 facilitates the exploitation of side-channel weaknesses.

implementations. A particular threat arises from asynchronous attacks, where the attacker only has to execute a program concurrently with the victim's program (on the same physical CPU) in order to collect temporal information about the victim's behavior. With this temporal information at hand, the attacker can reconstruct the internal workings of the victim. Because microarchitectural attacks execute on the victim as the victim, the attacker can only observe the victim's execution timings if the victim's execution cycles are not separated by ten target key-dependent operations or more.

...hemes today, ds, in particular ae (known as the a. It is known that is can in principle / is then a particular number problem (HNP). exploited in many attacks plementation defects or oncrete ECDSA implemen- s far have relied on at least 2

ephemeral random value called *nonce*, which is particularly sensitive: it is crucial to make sure that the nonces are kept in secret and sampled from the uniform distribution over a certain integer interval. It is easy to see that if the nonce is exposed or reused completely, then an attacker is able to extract the secret signing key by observing only a few signatures. By extending this simple observation, cryptanalysts have discovered stronger attacks that make it possible to recover the secret key even if short bit substrings of the nonces are leaked or biased. These extended attacks relate key recovery to the so-called hidden number problem (HNP) of Boneh and Venkatesan [15], and are part of a line of research initiated by Howgrave-Graham and Smart [36], who described a lattice-based

PARASITE: Password Recovery Attack against Srp presentations in The wild

Pierre-Alain Fouque
pa.fouque@gmail.com
Univ Rennes, CNRS, IRISA
Rennes, France

Mohamed Sabt
mohamed.sabt@irisa.fr
Univ Rennes, CNRS, IRISA
Rennes, France

KEYWORDS

SRP (PAKE)
password to
g key. The
offline
numerates

SRP; PAKE; Flush+Reload; PDA; OpenSSL; micro-architectural attack

ACM Reference Format:

Daniel De Almeida Braga, Pierre-Alain Fouque, and Mohamed Sabt. 2021.

Side-Channel Analysis of SM2: A Late-Stage Featurization Case Study

Nicola Tuveri
Tampere University of Technology
Tampere, Finland
nicola.tuveri@tut.fi

Cesar Pereida García
Tampere University of Technology
Tampere, Finland

Sohaib ul Hassan
Tampere University of Technology
Tampere, Finland
sohaibulhassan@tut.fi

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland

ABSTRACT
In recent years, the use of...

In recent years, applications increasingly adopt security primitives designed with better countermeasures against side channel attacks. A concrete example is LibCrypt's implementation of ECDH encryption with Curve25519. The implementation of ECDH Montgomery ladder scalar-by-point multiplication employs the unified, branchless Montgomery double-and-add formula and implements a constant-time argument swap within the ladder. However, LibCrypt's field arithmetic operations are not implemented in a constant-time side-channel-resistant fashion.

Based on the secure design of Curve25519, users of the curve are advised that there is no need to perform validation of input points. In this work we demonstrate that when this recommendation is followed, the mathematical structure of Curve25519 is exploited to exploit side-channel weaknesses.

On the secure design of Curve25519, users of the curve are advised that there is no need to perform validation of input points. In this work we demonstrate that when this recommendation is followed, the mathematical structure of Curve25519 facilitates the exploitation of side-channel weaknesses.

Luke Valenta
University of Pennsylvania
lukev@cis.upenn.edu

implementations. A particular threat arises from attacks, where the attacker only has to execute the victim's program (or the code) in order to collect temporal information about the internal workings of the victim.

Because microarchitectural attacks execute as part of the victim's program, the attacker can only observe the execution of the victim's program. This is a significant limitation, as the attacker can only observe the execution of the victim's program. This is a significant limitation, as the attacker can only observe the execution of the victim's program.

PA

Abstract

We demonstrate that the format in which private keys are persisted impacts Side Channel Analysis (SCA) security. Surveying several widely deployed software libraries, we investigate the formats they support, how they parse these keys, and what weaknesses and vulnerabilities, in extreme cases inducing entirely disjoint multi-precision arithmetic stacks deep in the cryptosystem level for keys that otherwise seem fully equivalent. Exploiting these vulnerabilities, we demonstrate key recovery attacks utilizing signals from electromagnetic (EM) emanations, to granularly reconstruct the private key from the format. We demonstrate the attack on a variety of formats, including PEM, PKCS#8, and DER. We show that the attack can be performed on a variety of hardware, including a variety of microcontrollers, and a variety of software, including a variety of operating systems. We show that the attack can be performed on a variety of hardware, including a variety of microcontrollers, and a variety of software, including a variety of operating systems.

Abstract

Certified Side Channels

Side Channels
Cesar Pereida García¹, Sohaib ul Hassan¹, Nicola Tuveri¹,
Iaroslav Gridin¹, Alejandro Cabrera Aldaya^{1,2}, and Billy Bob Brumley¹,
¹Tampere University, Tampere, Finland
²Universidad Tecnológica de la Habana (CUJAE), Habana, Cuba
cesarpereidagarcia.n.sohaibulhassan.nicola.tuveri. iaroslav.gridin, billy.brumley}@tuni.fi
aldaya@gmail.com

Abstract
Side Channel Analysis (SCA) is a technique used to extract sensitive information from cryptographic devices by analyzing the power consumption of the device during its operation. In this paper, we present a new SCA attack on the AES algorithm, which is based on the analysis of the power consumption of the device during the execution of the AES algorithm. The attack is able to extract the secret key of the device, which is a significant improvement over previous SCA attacks on AES. The attack is implemented on a hardware device, and the results show that the attack is successful in extracting the secret key of the device.

the multitude of standardized cryptographic key formats to choose from when persisting keys: *which one to choose, and does the choice matter?* Surprisingly, it does—we demonstrate different key formats trigger different behavior within software libraries, permeating all the way down to the low level arithmetic for the corresponding cryptographic primitive.

(ii) At the specification level, alongside required parameters, standardized key formats often contain optional parameters, *does including or excluding optional parameters impact security?* Surprisingly, it does. We demonstrate that omitting optional parameters can cause extremely different execution flows deep within a software library, and also that *seemingly mathematically identical* algorithms can

ACM Reference Format

Daniel De Almeida Braga, Pierre-Alain Michel et al.

Akira Takahashi
DIGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Yuval Yarom
University of Adelaide and Data61
Australia
yuval@cs.adelaide.edu.au

-shi

Diego F. Aranha
DIGIT, Aarhus University
Denmark
dfaranha@eng.au.dk

Yuval
University of Adelaide
yval@cs.adelaide.edu.au

Nicola Tuveri
Tampere University of Technology
Tampere, Finland
nicola.tuveri@tut.fi

Cesar Pereida García
Tampere University of Technology
Tampere, Finland

Sohaib ul Hassan
Tampere University of Technology
Tampere, Finland
sohaibulhassan@tut.fi

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland

ACM Reference Format

Daniel De Almeida Braga, Pierre-Alain Michel et al.



**May the Fourth Be With You: A Microarray
Attack on Several Real-World Applications**

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

Daniel Genkin
University of Pennsylvania and
University of Maryland
danielg3@cis.upenn.edu

ABSTRACT

In recent years, applications increasingly designed with better countermeasures. A concrete example is Libsolv encryption with Curve25519. Libsolv, branchless Montgomery ladder, and Libsolv are constant-time.

**Pseudorandom Black Swans:
Cache Attacks on CTR_DRBG**

Shaanan Coeney¹, Andrew Kwong², Shahar Paz³, Daniel Genkin², Nudra
¹University of Michigan, {ankwong-genkin}@umich.edu
²University of Michigan, shaharpaz@tau.ac.il
³Tel Aviv University, shaharpaz@tau.ac.il
 of California, COSIC (KU Leuven), er@cybrol.net
 vval@cs.adelaida.edu.au

¹University of Michigan, {ankwong-genkin,
shaharps@tau.ac.il}

²University of Michigan, {unkwong@tatu.umd.edu, shaharps@tatu.umd.edu}

³Tel Aviv University, and COSIC (KU Leuven), er@cs.tau.ac.il, yval@cs.adelaide.edu.au

Abstract—Modern cryptography requires the ability to securely generate pseudorandom numbers. However, despite decades of work on side-channel attacks, there is little discussion of their application to pseudorandom number generators (PRGs). In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRG implementations. We show that hard-learned lessons about side channel leakage are not being used nor been applied to PRGs, at all levels of security. We recommend that designers consider side channel resistance as a design level requirement for cryptographic hardware.

We find that hard-learned lessons from encryption primitives have not been applied to the design level, the levels of abstraction. At the design level, the CTR_DRBG design does not have forward security attack. At the levels of abstraction, the CTR_DRBG design does not have forward security attack. At the levels of abstraction, the CTR_DRBG design does not have forward security attack. At the levels of abstraction, the CTR_DRBG design does not have forward security attack.

The simplest theoretical PRG construction is an algorithm that expands a smaller seed into a longer output sequence that is computationally indistinguishable from a true sequence of random bits. However, the practical security demands for random number generation are somewhat more complex; in real systems, these pseudorandom number generator constructions are often multi-stage algorithms that collect inputs from environmental entropy sources or hardware into an "entropy pool". The pool is then used to seed a PRG that generates cryptographically secure output. Real world PRGs must also meet additional security guarantees, including recovery from state compromise.

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland

LadderLeak: Breaking ECDSA With Less Than One Bit Of Nonce Leakage

Diego F. Aranha
DIGIT, Aarhus University
Denmark
dfaranha@... .

Felipe Rodrigues Novaes
University of Campinas
Brazil
ra135663@students.ic.unicamp.br

Yuval Yarom
University of Adelaide and Data61
Australia
yarom@cs.adelaide.edu.au

Akira Takahashi
DIGIT, Aarhus University
Denmark
takahashi@cs.au.dk

Certified Side Channels

Abstract

Abstract

format in which private keys are per-
Analysis (SCA) security. Survey-
software libraries, we investigate
we parse these keys, and what
to uncover a combination of
to extreme cases inducing
arithmetic stacks deep
that otherwise seem
vulnerabilities, we de-
very attacks utilizing signals
emanations, to granular
age (PASS-
password to
key. The
list offline
numerates

ACM Reference Format

ACM Reference Format:
Daniel De Almeida Braga, Pierre-Alain Huet. 2018. *Formalizing the Theory of the Lambda Calculus in Coq*. In *CADE*. 1–17.

the multitude of standardized cryptographic key formats to choose from when persisting keys: *which one to choose, and does the choice matter?* Surprisingly, it does—we demonstrate different key formats trigger different behavior within software libraries, permeating all the way down to the low level arithmetic for the corresponding cryptographic primitive.

(ii) At the specification level, alongside required parameters, standardized key formats often contain optional parameters, *does including or excluding optional parameters impact security?* Surprisingly, it does. We demonstrate that omitting optional parameters can cause extremely different execution flows deep within a software library, and also that *seemingly innocuous mathematical operations* can have a significant impact on the execution of cryptographic algorithms.

Reference Format:
De Almeida Braga, Pierre-Alain Fouquet, and Jean-Luc Gauthier. 2023. *Standardized Cryptographic Key Formats: A Systematic Study of Their Impact on Cryptographic Libraries*. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS '23)*. ACM, New York, NY, 1–15.



Des attaques partout...

+ CVE-2005-0109, CVE-2013-4242, CVE-2014-0076, CVE-2016-0702, CVE-2016-2178, CVE-2016-7440, CVE-2016-7439, CVE-2016-7438, CVE-2018-0495, CVE-2018-0737, CVE-2018-10846, CVE-2019-9495, CVE-2019-13627, CVE-2019-13628, CVE-2019-13629, CVE-2020-16150, CVE-2020-36421, CVE-2023-5388, CVE-2023-6135, CVE-2024-37880 ...

LadderLeak: Breaking ECDSA

cularly sensi-
cept in secret
ertain integer
sed or reused
-using key



Shaanan Cooney¹, ...
²University of ...
³Tel Aviv University, San Diego, ...
⁴University of California, San Diego, ...
⁵Tel Aviv University and COSIC (KU Leuven), ...
⁶University of Adelaide and Data61, yval@cs.adelaide.edu.au

Abstract—Modern cryptography requires the ability to securely generate pseudorandom numbers. However, despite decades of work on side-channel attacks, there is little discussion of their application to pseudorandom number generators (PRGs). In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRG implementations. We find that hard-learned lessons have not been applied to PRGs, at all levels of abstraction. At the design level, the NIST-recommended CTR_DRBG design does not have forward security if an attacker is able to compromise the state via a side-channel attack. At the implementation level, popular implementations of CTR_DRBG such as the FIPS module and NetBSD's kernel use leaky T-DES and ECB block cipher, enabling cache side-

The simplest theoretical PRG construction is an algorithm that expands a smaller seed into a longer output sequence that is computationally indistinguishable from a true sequence of random bits. However, the practical security demands for random number generation are somewhat more complex: in real systems, these pseudorandom number generator constructions are often multi-stage algorithms that collect inputs from environmental entropy sources or hardware into an “entropy pool”. The pool is then used to seed a PRG that generates cryptographically secure output. Real world PRGs must also meet additional security guarantees, including recovery from state compromise.

Abstract

Format in which private keys are per-
software libraries, we investigate
key parse these keys, and what
we uncover a combination of
extreme cases inducing
arithmetic stacks deep
that otherwise seem
vulnerabilities, we de-
very attacks utilizing signals
-etic (EM) emanations, to granular
ge (PAN)
ssword to
g key. The
sist offline
numerates

de la Habana (CUJAE), Habana, Cuba
aldaya@gmail.com
...brumley!
...@tuni.fi

the multitude of standardized cryptographic key formats to choose from when persisting keys: which one to choose, and does the choice matter? Surprisingly, it does—we demonstrate different key formats trigger different behavior within software libraries, permitting all the way down to the low level arithmetic for the corresponding cryptographic primitive. (ii) At the specification level, alongside required parameters, standardized key formats often contain optional parameters, does including or excluding optional parameters impact security? Surprisingly, it does. We demonstrate that omitting optional parameters can cause extremely different execution flows deep within a software library, and also that weeminally mathematically identical at the

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland
billy.bob@tut.fi

ACM Reference Format:
Daniel De Almeida Braga, Pierre-Alain Fouquet, and Billy Bob Brumley.
LadderLeak: Breaking ECDSA. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2024.

Des attaques partout...

+ CVE-2005-0109, CVE-2013-4242, CVE-2014-0076, CVE-2016-0702, CVE-2016-2178, CVE-2016-7440, CVE-2016-7439, CVE-2016-7438, CVE-2018-0495, CVE-2018-0737, CVE-2018-10846, CVE-2019-9495, CVE-2019-13627, CVE-2019-13628, CVE-2019-13629, CVE-2020-16150, CVE-2020-36421, CVE-2023-5388, CVE-2023-6135, CVE-2024-37880 ...

LadderLeak: Breaking ECDSA

cularly sensi-
cept in secret
tain integer
sed or reused
-ring key



Shaanan Cooney¹, ...
¹University of ...
²Tel Aviv University, San Diego, ...
University of California, San Diego, ...
and COSIC (KU Leuven), ...
yval@cs.adelaida.edu.au

struction is an algorithm
-out sequence

Abstract

de la Habana (CUJAE), Habana, Cuba
aldaya@gmail.com
...brumley!
...@tuni.fi

Des. Attaques. Partout.

Abstract—Modern cryptographic systems rely on pseudorandom number generators (PRNGs) to generate pseudorandom numbers. Decades of work on side-channel attacks, empirically evaluating their application to pseudorandom number generators, have shown that side-channel leakage is a real threat. In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRNGs, at all levels of abstraction. At the design level, the NIST-recommended CTR_DRBG design does not have forward security if an attacker is able to compromise the state via a side-channel attack. At the implementation level, popular implementations of CTR_DRBG, such as the FIPS module and NetBSD's kernel use leaky T-DES, enabling block cipher, enabling cache side-

random systems, these are often multi-stage algorithms. Real world PRNGs must also be cryptographically secure output. Real world PRNGs must also meet additional security guarantees, including recovery from state compromise.

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland
bbrumley@tut.fi

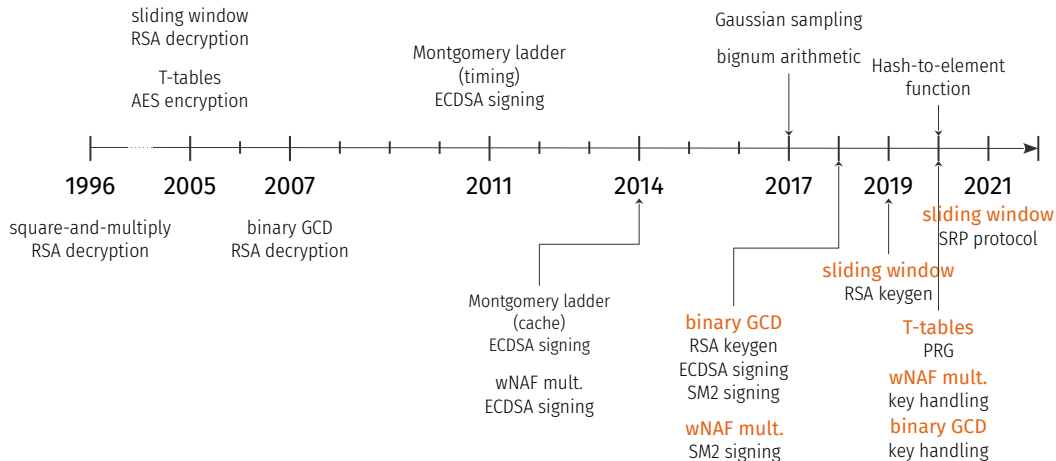
Abstract—Modern cryptographic systems rely on pseudorandom number generators (PRNGs) to generate pseudorandom numbers. Decades of work on side-channel attacks, empirically evaluating their application to pseudorandom number generators, have shown that side-channel leakage is a real threat. In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRNGs, at all levels of abstraction. At the design level, the NIST-recommended CTR_DRBG design does not have forward security if an attacker is able to compromise the state via a side-channel attack. At the implementation level, popular implementations of CTR_DRBG, such as the FIPS module and NetBSD's kernel use leaky T-DES, enabling block cipher, enabling cache side-

Abstract—Modern cryptographic systems rely on pseudorandom number generators (PRNGs) to generate pseudorandom numbers. Decades of work on side-channel attacks, empirically evaluating their application to pseudorandom number generators, have shown that side-channel leakage is a real threat. In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRNGs, at all levels of abstraction. At the design level, the NIST-recommended CTR_DRBG design does not have forward security if an attacker is able to compromise the state via a side-channel attack. At the implementation level, popular implementations of CTR_DRBG, such as the FIPS module and NetBSD's kernel use leaky T-DES, enabling block cipher, enabling cache side-

ACM Reference Format:

Daniel De Almeida Braga, Pierre-Alain Fouquet, and Billy Bob Brumley.

Comparaison de vulnérabilités récentes (2017-2022) et anciennes



Les mêmes vulnérabilités reviennent à la surface, pourquoi ?

Nouveaux contextes

- génération de clés
 - traitement des clés
 - génération de nombres aléatoires
-
- le code vulnérable reste dans la bibliothèque (surtout pour OpenSSL)

Les mêmes vulnérabilités reviennent à la surface, pourquoi ?

Nouveaux contextes

- génération de clés
- traitement des clés
- génération de nombres aléatoires

Nouvelles bibliothèques

- implémentation RSA "sliding window" dans **MbedTLS** (2017)
- attaques Bleichenbacher dans **MbedTLS, s2n, et NSS** (2019)

- le code vulnérable reste dans la bibliothèque (surtout pour OpenSSL)
- vulnérabilités trouvées dans OpenSSL mais patchs non propagés aux autres bibliothèques

La plupart des vulnérabilités proviennent de code déjà identifié comme étant vulnérable

Beaucoup d'outils (académiques) existants, mais

- peu connus, peu utilisés, peu utilisables
- toujours des limitations techniques
- des architectures de bibliothèques cryptographiques très complexes

Conclusions

- papier séminal par Kocher en 1996 : près de 30 ans de recherche
- domaine en expansion : forte augmentation du nombre d'articles depuis 2015
- les attaques microarchitecturales nécessitent :
 - la compréhension et le contrôle bas niveau du matériel → microarchitecture, rétro-ingénierie
 - la compréhension très fine du logiciel → analyse de programme, compilation, cryptographie

→ il y a du travail dans toutes les couches d'abstraction !

Merci !

Contact

✉ `clementine.maurice@cnrs.fr`

🐦 @BloodyTangerine / @bloody-tangerine.bsky.social

Cybersécurité à la loupe : du logiciel au matériel

Clémentine Maurice, Chargée de recherche CNRS

Laboratoire CRIS^tAL, équipe Spirals

1er Octobre 2025—Bar des sciences, Université de Lille