KU LEUVEN

Partial Key Overwrite Attacks in Microcontrollers: a Survey

CASCADE 2025

pcy Sluys, Lennert Wouters, Benedikt Gierlichs, Ingrid Verbauwhede

COSIC-KU Leuven

2025-04-04

Outline

KU LEUVEN

Outline

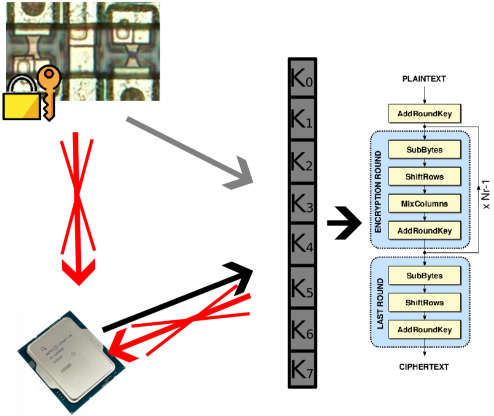
What are PKO attacks?

- Chips often have cryptographic accelerators[Citation Needed]
- In some implementations, the key is kept separate from the CPU
  - PUF
  - OTP/fuses
  - ...
  - Hardware KDF
  - TEE

What are PKO attacks?

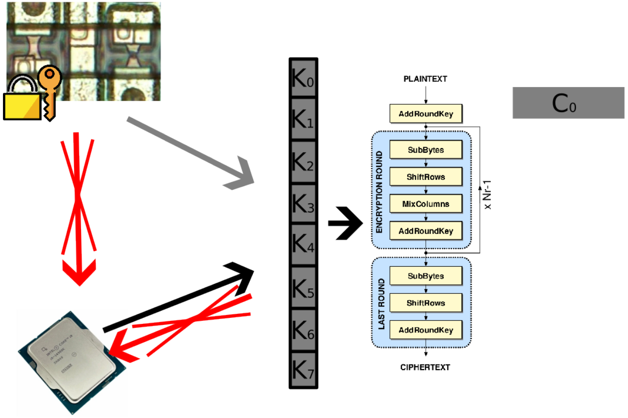
- Chips often have cryptographic accelerators[Citation Needed]
- In some implementations, the key is kept separate from the CPU
  - PUF
  - OTP/fuses
  - ...
  - Hardware KDF
  - TEE
- If **partial** overwrite of a **write-only** key register allowed: key leakage!
  - $\Rightarrow$ **P**artial **K**ey **O**verwrite attack

An example

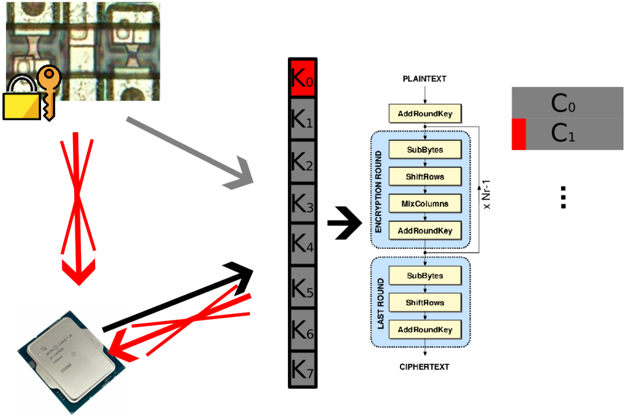


K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT

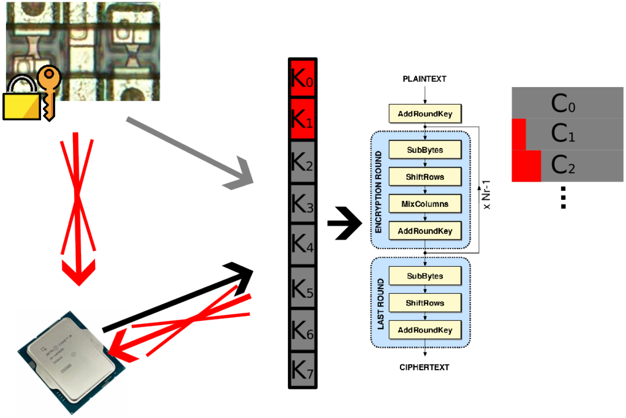
An example


K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
C0

An example

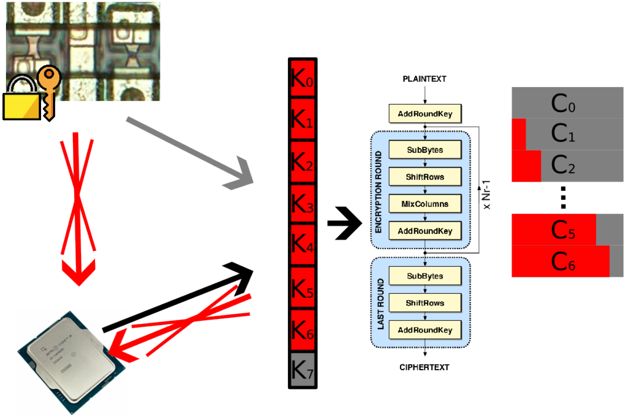

K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
C0
C1
⋮

An example

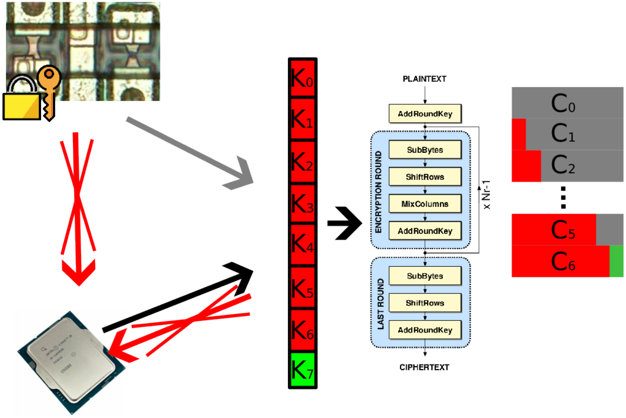


KU LEUVEN

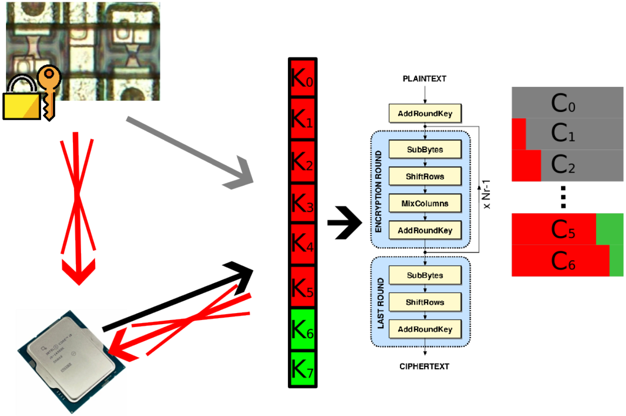
An example


K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
C0
C1
C2
C5
C6

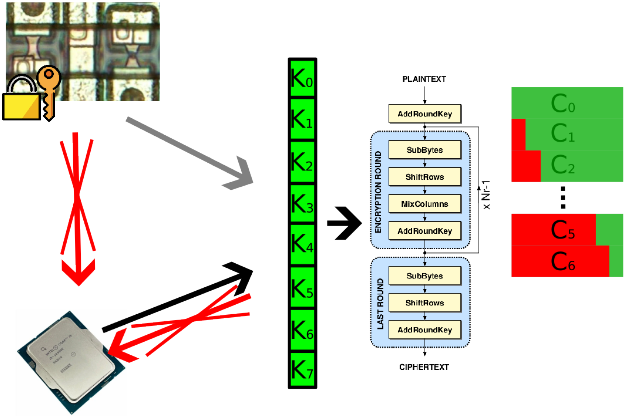
An example


K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
C0
C1
C2
C5
C6

An example

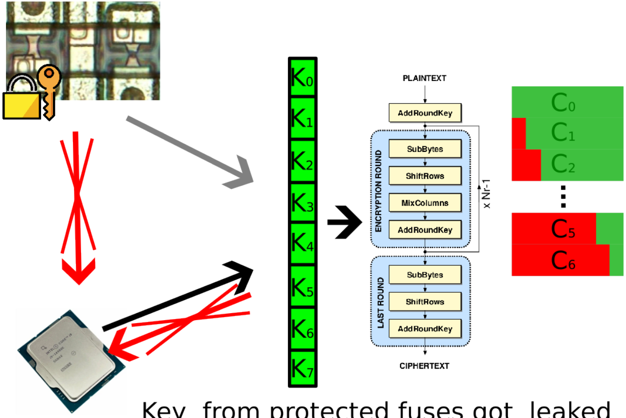

K0
K1
K2
K3
K4
K5
K6
K7
PLAINTEXT
AddRoundKey
ENCRYPTION ROUND
SubBytes
ShiftRows
MixColumns
AddRoundKey
x Nr-1
LAST ROUND
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
C0
C1
C2
C5
C6

An example



KU LEUVEN

An example



Key from protected fuses got leaked

Outline

Attacker model

- Attacker can query cryptographic module
- Key not exposed to attacker
- Attacker can overwrite **parts of** the key

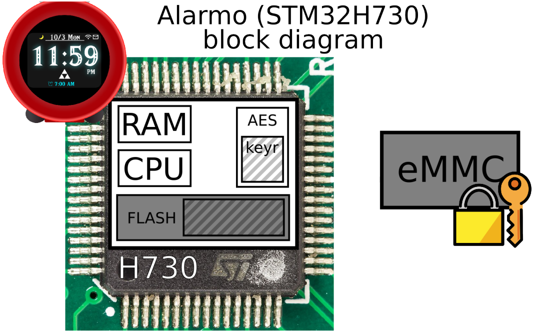
Attacker model

- Attacker can query cryptographic module
- Key not exposed to attacker
- Attacker can overwrite **parts of** the key

Embedded context:

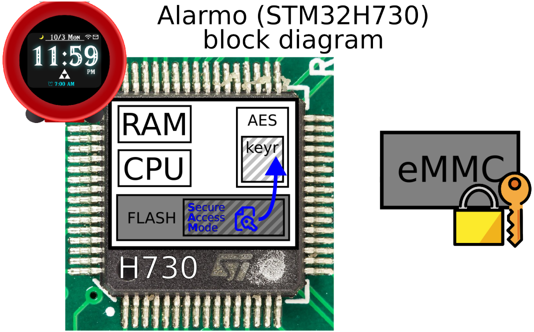
- Query cryptographic module using low-privilege code execution or debug access
- Key secured using TEE, protected fuses, boot ROM, ...

Example: Alarmo[1]



[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

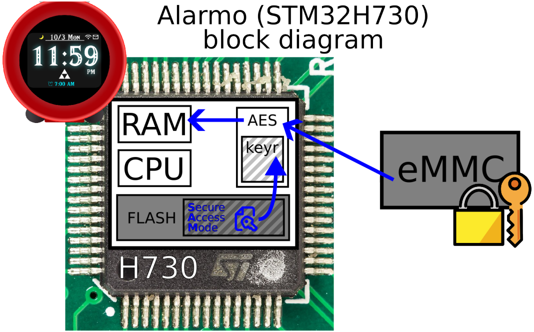
Example: Alarmo[1]



[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

Example: Alarmo[1]

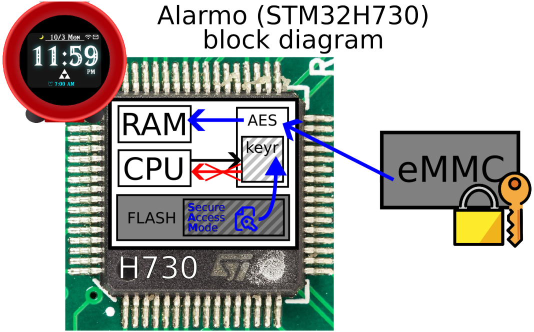
Alarmo (STM32H730)
block diagram



Based on CC BY-SA 4.0 images by Raimond Spekking and PantheraLeo1359531

[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

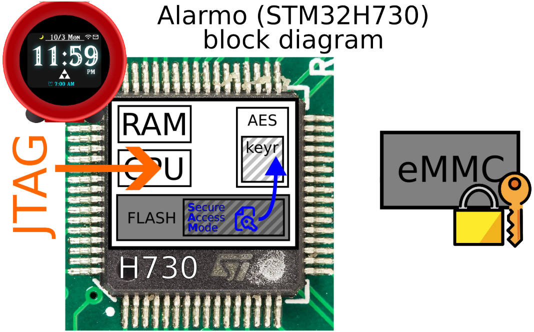
Example: Alarmo[1]



Based on CC BY-SA 4.0 images by Raimond Spekking and PantheraLeo1359531

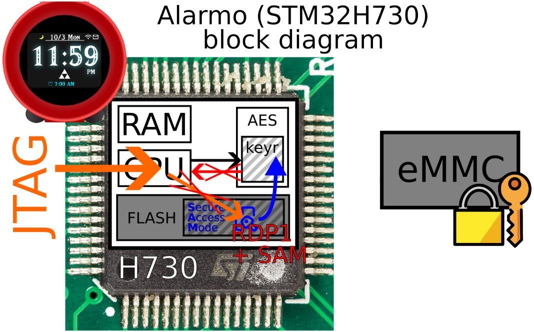
[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

Example: Alarmo[1]



[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

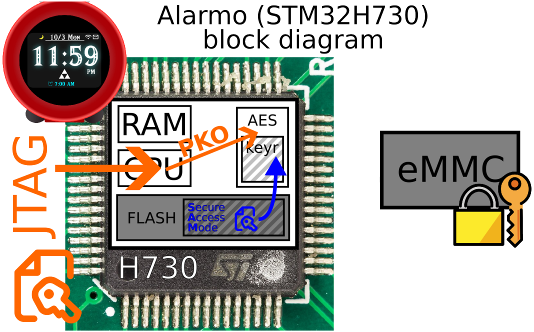
Example: Alarmo[1]



Based on CC BY-SA 4.0 images by Raimond Spekking and PantheraLeo1359531

[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

Example: Alarmo[1]



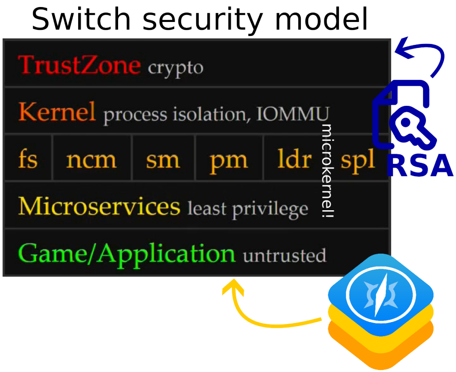
[1] https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html

Example: Nintendo Switch[2]

Switch security model

[2] https://switchbrew.org/wiki/Switch_System_Flaws#TrustZone

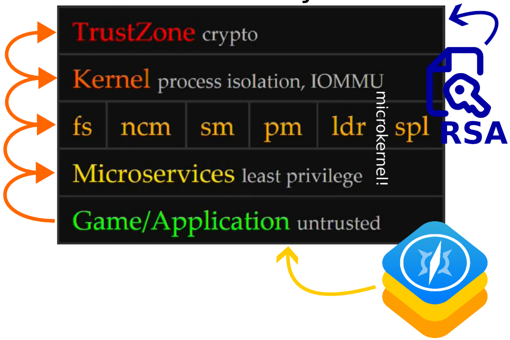
Example: Nintendo Switch[2]



[2] https://switchbrew.org/wiki/Switch_System_Flaws#TrustZone

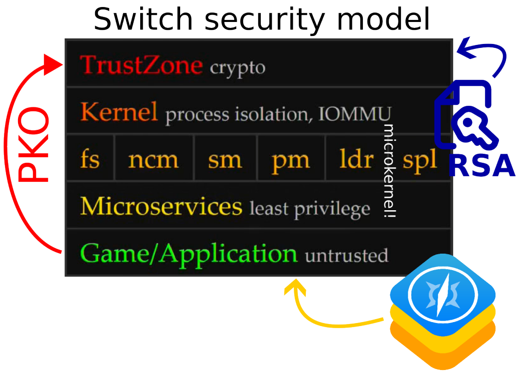
Example: Nintendo Switch[2]

Switch security model



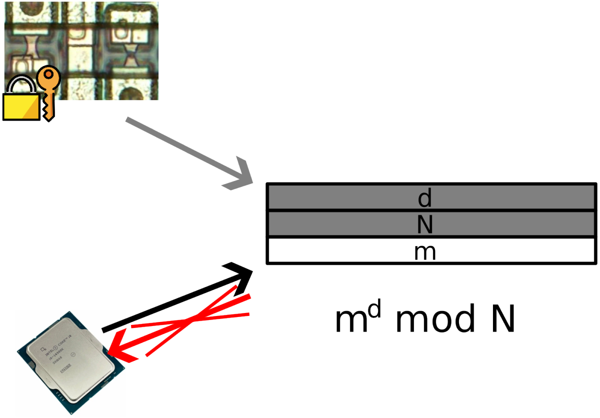
[2] https://switchbrew.org/wiki/Switch_System_Flaws#TrustZone

Example: Nintendo Switch[2]

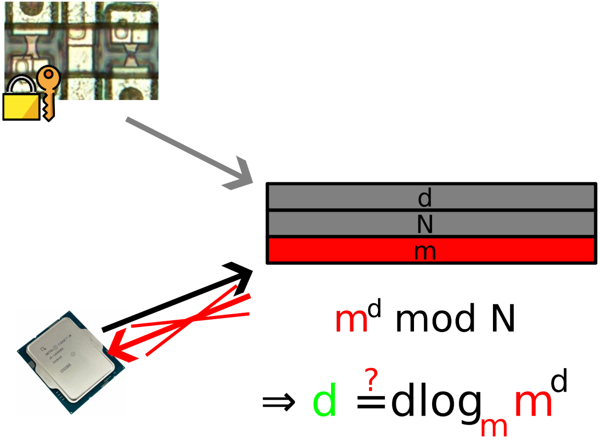


[2] https://switchbrew.org/wiki/Switch_System_Flaws#TrustZone

Not just block ciphers



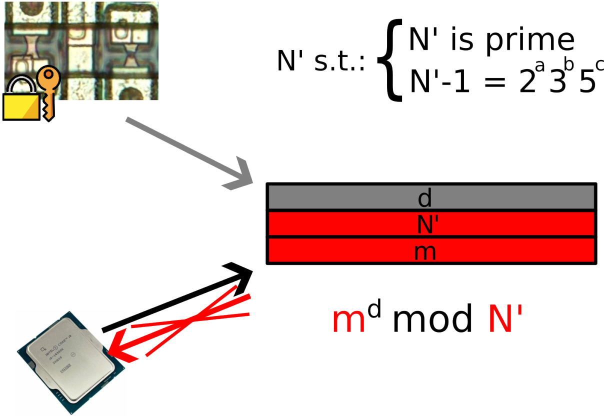
Not just block ciphers



$m^d \bmod N$

$\Rightarrow d \stackrel{?}{=} \mathrm{dlog}_m m^d$

Not just block ciphers



Not just block ciphers

N' s.t.: $\begin{cases} N' \text{ is prime} \\ N'-1 = 2^a 3^b 5^c \end{cases}$

⇒ Pohlig-Hellman to calculate dlog

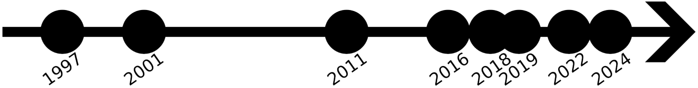
d
N'
m

$m^d \bmod N'$

⇒ $d = \text{dlog}_m \, m^d \bmod N'$

KU LEUVEN

Timeline of PKO attacks



KU LEUVEN

Timeline of PKO attacks

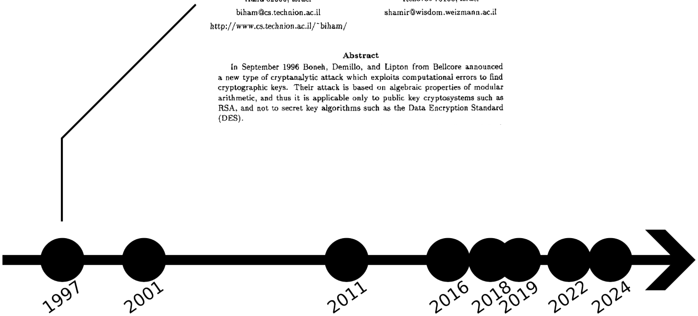
Differential Fault Analysis of Secret Key Cryptosystems

Eli Biham
Computer Science Department
Technion – Israel Institute of Technology
Haifa 32000, Israel
biham@cs.technion.ac.il
http://www.cs.technion.ac.il/~biham/

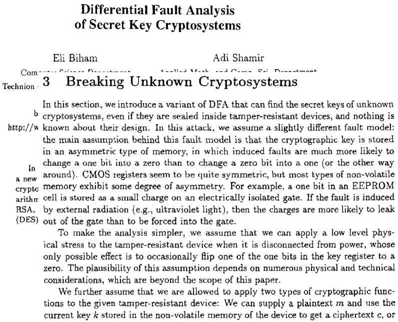
Adi Shamir
Applied Math. and Comp. Sci. Department
The Weizmann Institute of Science
Rehovot 76100, Israel
shamir@wisdom.weizmann.ac.il

Abstract

In September 1996 Boneh, Demillo, and Lipton from Bellcore announced a new type of cryptanalytic attack which exploits computational errors to find cryptographic keys. Their attack is based on algebraic properties of modular arithmetic, and thus it is applicable only to public key cryptosystems such as RSA, and not to secret key algorithms such as the Data Encryption Standard (DES).



Timeline of PKO attacks



Differential Fault Analysis of Secret Key Cryptosystems

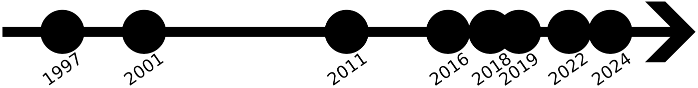
Eli Biham   Adi Shamir

3 Breaking Unknown Cryptosystems

In this section, we introduce a variant of DFA that can find the secret keys of unknown cryptosystems, even if they are sealed inside tamper-resistant devices, and nothing is known about their design. In this attack, we assume a slightly different fault model: the main assumption behind this fault model is that the cryptographic key is stored in an asymmetric type of memory, in which induced faults are much more likely to change a one bit into a zero than to change a zero bit into a one (or the other way around). CMOS registers seem to be quite symmetric, but most types of non-volatile memory exhibit some degree of asymmetry. For example, a one bit in an EEPROM cell is stored as a small charge on an electrically isolated gate. If the fault is induced by external radiation (e.g., ultraviolet light), then the charges are more likely to leak out of the gate than to be forced into the gate.

To make the analysis simpler, we assume that we can apply a low level physical stress to the tamper-resistant device when it is disconnected from power, whose only possible effect is to occasionally flip one of the one bits in the key register to a zero. The plausibility of this assumption depends on numerous physical and technical considerations, which are beyond the scope of this paper.

We further assume that we are allowed to apply two types of cryptographic functions to the given tamper-resistant device: We can supply a plaintext m and use the current key k stored in the non-volatile memory of the device to get a ciphertext c, or

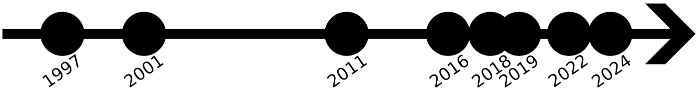


Timeline of PKO attacks

Attack on Private Signature Keys of the OpenPGP Format, PGP(TM) Programs and Other Applications Compatible with OpenPGP

Vlastimil Klima and Tomas Rosa

cipher nor user´s secret passphrase. A modification of the private key file in a certain manner and subsequent capturing of one signed message is sufficient for successful attack. Insufficient protection of the



KU LEUVEN

Timeline of PKO attacks

DSi photo © Evan Amos, CC BY-SA 3.0

Yellows8 2011

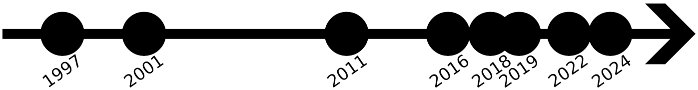
AES_Engine allows partial key overwrite

After using the key generator to generate the normal-key, you could overwrite parts of the normal-key with your own data and then recover the key-generator output by comparing the new crypto output with the original crypto output. From the normal-key outputs, you could deduce the key-generator function.

This applies to the keyX/keyY too.

The 3DS TWL AES engine is also affected.

https://dsibrew.org/wiki/DSi_system_flaws



Timeline of PKO attacks


debugmode
Projects, hardware fun and everything between it.

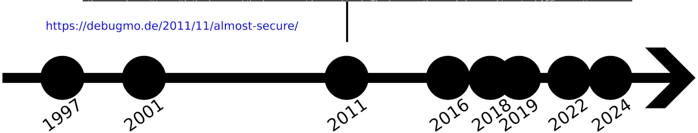
2011-11-01

Almost Secure

5. Write-Only Registers with Partial Override allow Key Recovery

This one is a classic as well. Some hardware implements cryptographic functions (for example AES decryption) in hardware. S
hardware designers cleverly make the registers that are used to configure the key used in the decryption write only - such as

https://debugmo.de/2011/11/almost-secure/



KU LEUVEN

Timeline of PKO attacks

Myria, 2016

> RSA keyslots don't clear exponent when setting modulus

The RSA keyslots are set by boot ROM to have four private RSA keys. The exponent value in the RSA registers is write-only and not readable.

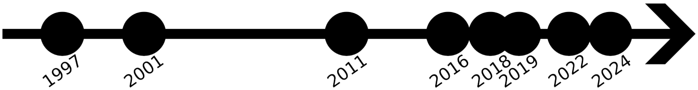
However, when setting a keyslot's modulus, the RSA hardware leaves the exponent alone. This allows retrieving the exponent by doing a discrete logarithm of the output.

By setting the modulus to a prime number whose modular multiplicative order is "smooth" (that is, p-1 is divisible by only small prime numbers), discrete logarithms can be calculated quickly using the Pohlig-Hellman algorithm. If the prime chosen is greater than the modulus, but the same bit size, the discrete logarithm is the private exponent.

This exploit's usefulness is limited: RSA keyslot 0 is only used in current firmware for deriving the 6.x save and 7.x NCCH keys, which were already known, and the other three keyslots are entirely unused. Additionally, with a boot ROM dump, this exploit is moot; these private keys are located in the protected ARM9 boot ROM.

https://www.3dbrew.org/wiki/3DS_System_Flaws

3DS photo © Evan Amos, PD



Timeline of PKO attacks

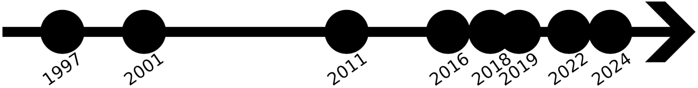
Yifan Lu

RANDOM STUFF I'M MAKING AND THINKING

The 3DS Cryptosystem

gone. What about content protection? It's hanging by its last thread. We can always use the 3DS as a blackbox for decrypting content. However, the goal for attackers is to get the "normal" keys and be able to decrypt content offline. The key generator is the defense for this. Remember that security by obscurity can only buy you so much time? It took about four years since the original release of the 3DS for hackers to break it.

https://yifan.lu/2016/04/06/the-3ds-cryptosystem/



KU LEUVEN

Timeline of PKO attacks

Security Engine keyslots vulnerable to partial overwrite attack

SciresM, 2018

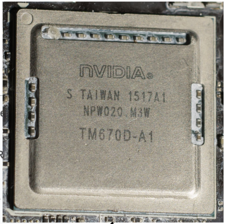
Tegra X1

The Tegra X1 security engine supports writing ke
to the engine with syntax as follows:

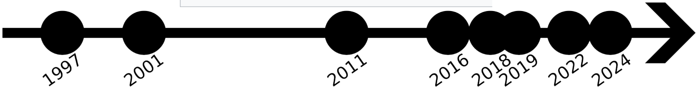
SECURITY_ENGINE->AES_KEYTABLE_ADDR
<< 4) | (dword_index_in_keyslot);

SECURITY_ENGINE->AES_KEYTABLE_DATA
readle32(key, dword_index_in_keyslot * 4);

However, the Security Engine flushes writes to the internal key tables immediately when AES_KEYTABLE_DATA is written -- this allows one to overwrite a single dword of a key at a time, and thus brute force the contents of keyslots in time (2^32 * 8) = 2^35 instead of 2^256.



https://switchbrew.org/wiki/Switch_System_Flaws



Timeline of PKO attacks

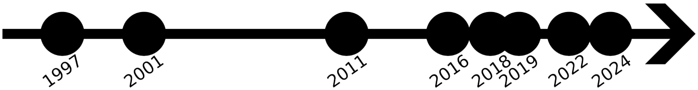
Fault Attacks on Secure Embedded Software: Threats, Design and Evaluation

Bilgiday Yuce
Virginia Tech
Blacksburg, VA
bilgiday@vt.edu

Patrick Schaumont
Virginia Tech
Blacksburg, VA
schaum@vt.edu

Marc Witteman
Riscure – Security Lab
Delft, Netherlands
witteman@riscure.com

original secret value can be detected. For this reason, for example, write-only cryptographic key registers should never allow partial update, otherwise the attacker can test a partial key guess by detecting these collisions.



KU LEUVEN

Timeline of PKO attacks

TrustZone allows using imported RSA exponents with arbitrary modulus

SciresM, 2019

TrustZone supports "importing" RSA priva
encrypted with TrustZone only keydata in
console that has compromised userland f
calculations with them offline. In practice
and SSL (console client cert communica

However, the actual SMC API only impo
separately by userland in each call. The
that userland can pass in any message
private exponent) % modulus back from the

By choosing a prime number modulus P such that P has "smooth" order (totient(P)
only by "small" primes), one can efficiently use the Pohlig-Hellman algorithm to calculate the discrete logarithm of such a result directly, and thus obtain the private exponent.

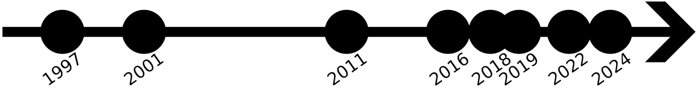
This is mostly useless in practice, given the general availability of other exploits to obtain these decrypted exponents.

This was fixed in 10.0.0 by importing the modulus in addition to the exponent for the ES device key and ES client cert key. For backwards compatibility reasons the SSL key and Lotus key still only import the exponent.

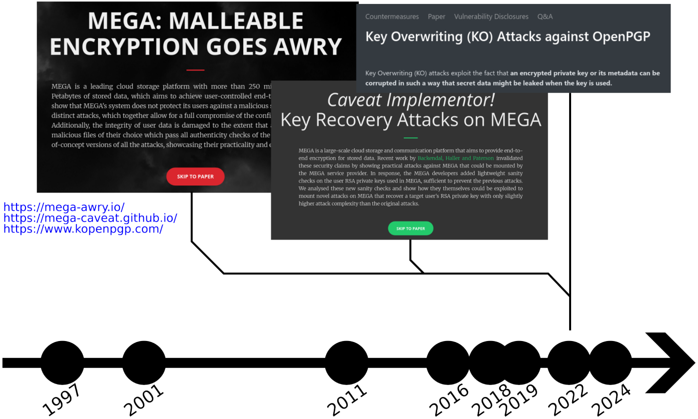
StorageExpMod also now validates that the exponentiation of "DDDDD..." about the provided modulus by the imported exponent and then the fixed public exponent returns "DDDDD...", and returns invalid argument if validation fails.

Switch photo © Evan Amos, PD

https://switchbrew.org/wiki/Switch_System_Flaws



Timeline of PKO attacks



https://mega-awry.io/
https://mega-caveat.github.io/
https://www.kopenpgp.com/

Timeline of PKO attacks

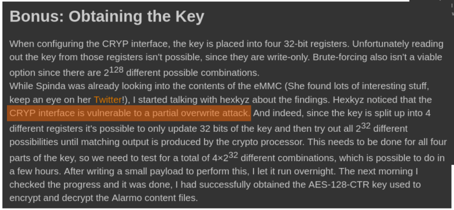

Gary's hacking stuff

Tuesday, October 29, 2024

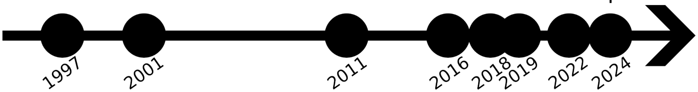
Looking into the Nintendo Alarmo

While everyone was waiting on news for the successor of the Nintendo Switch, Nintendo released the
l plastic alarm clock that can wake you up with sounds from your favorite Nintendo
was hesitant to buy one at first, I eventually decided to get one and look deeper into how

https://garyodernichts.blogspot.com/2024/10/looking-into-nintendo-alarmo.html


Bonus: Obtaining the Key

When configuring the CRYP interface, the key is placed into four 32-bit registers. Unfortunately reading out the key from those registers isn't possible, since they are write-only. Brute-forcing also isn't a viable option since there are 2^{128} different possible combinations.
While Spinda was already looking into the contents of the eMMC (She found lots of interesting stuff, keep an eye on her Twitter!), I started talking with hexkyz about the findings. Hexkyz noticed that the CRYP interface is vulnerable to a partial overwrite attack. And indeed, since the key is split up into 4 different registers it's possible to only update 32 bits of the key and then try out all 2^{32} different possibilities until matching output is produced by the crypto processor. This needs to be done for all four parts of the key, so we need to test for a total of 4×2^{32} different combinations, which is possible to do in a few hours. After writing a small payload to perform this, I let it run overnight. The next morning I checked the progress and it was done, I had successfully obtained the AES-128-CTR key used to encrypt and decrypt the Alarmo content files.



Timeline of PKO attacks

KU LEUVEN

Attack comparison and context

- Mounted using software exploits:
  - PGP attacks
  - MEGA attacks
- Mounted using invasive circuit-level attacks:
  - Partial EEPROM wipe in [1]
- Mounted using fault injection:
  - Safe Error Analysis

Attack comparison and context

- Mounted using software exploits:
  - PGP attacks
  - MEGA attacks
- Mounted using invasive circuit-level attacks:
  - Partial EEPROM wipe in [1]
- Mounted using fault injection:
  - Safe Error Analysis
- Mounted using **[???]**:
  - PKO in microcontroller

KU LEUVEN

"Mounted using [???]"?

Exact attack method?

- Not software vuln: attacking bug in **hardware** state machine
- Not side channel or fault attack
- Not an invasive attack
- Not a microarchitectural attack: **not only** in CPU!

"Mounted using [???]"?

Exact attack method?

- Not software vuln: attacking bug in **hardware** state machine
- Not side channel or fault attack
- Not an invasive attack
- Not a microarchitectural attack: **not only** in CPU!

Different attack:

- Hardware components work fine: AES, CPU, DMA, IRQ, system bus, timer, ...

"Mounted using [???]"?

Exact attack method?

- Not software vuln: attacking bug in **hardware** state machine
- Not side channel or fault attack
- Not an invasive attack
- Not a microarchitectural attack: **not only** in CPU!

Different attack:

- Hardware components work fine: AES, CPU, DMA, IRQ, system bus, timer, ...
- *Composition* of blocks $\Rightarrow$ *hardware* bugs

"Mounted using [???]"?

Exact attack method?

- Not software vuln: attacking bug in **hardware** state machine
- Not side channel or fault attack
- Not an invasive attack
- Not a microarchitectural attack: **not only** in CPU!

Different attack:

- Hardware components work fine: AES, CPU, DMA, IRQ, system bus, timer, ...
- *Composition* of blocks ⇒ *hardware* bugs

Examples:

- Flash readout protect circumvention using instruction fetches [6]
- 'Execute-only memory' readout using timer interrupts [3, 5]
- More: [2], [4, `ntrcardhax`], [7], [8]

“Mounted using [???]”?

Exact attack method?

- Not software vuln: attacking bug in **hardware** state machine
- Not side channel or fault attack
- Not an invasive attack
- Not a microarchitectural attack: **not only** in CPU!

Different attack:

- Hardware components work fine: AES, CPU, DMA, IRQ, system bus, timer, ...
- *Composition* of blocks $\Rightarrow$ *hardware* bugs

Examples:

- Flash readout protect circumvention using instruction fetches [6]
- ‘Execute-only memory’ readout using timer interrupts [3, 5]
- More: [2], [4, `ntrcardhax`], [7], [8]

Nameless...

"Mounted using [???]"?

- "Standalone components work fine"

E

.

gs

Nameless...

"Mounted using [???]"?

- "Standalone components work fine"
- "Interfacing of blocks is complex, bugs arise"

E

.

gs

Nameless...

"Mounted using [???]"?

- "Standalone components work fine"
- "Interfacing of blocks is complex, bugs arise"

(with help from people on Mastodon...)

VULNERABLE RESULT OF INDIVIDUALLY SECURE COMPONENTS ATTACK

E

.

gs

Nameless...

"Mounted using [???]"?

- "Standalone components work fine"
- "Interfacing of blocks is complex, bugs arise"

(with help from people on Mastodon...)

VULNERABLE **R**ESULT OF **I**NDIVIDUALLY **S**ECURE **K**OMPONENTS **A**TTACK

VRISKA (from *Homestuck*)

E... ...gs

Nameless...

Outline

This raises the question,

Is this still a problem nowadays?

How many off-the-shelf chips are vulnerable?

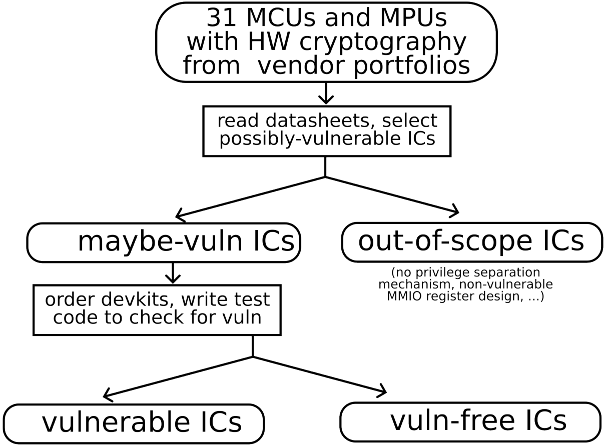
This raises the question,

Is this still a problem nowadays?

How many off-the-shelf chips are vulnerable?

$\Rightarrow$ **Time for a survey**

Method


31 MCUs and MPUs
with HW cryptography
from vendor portfolios
read datasheets, select
possibly-vulnerable ICs
maybe-vuln ICs
out-of-scope ICs
(no privilege separation
mechanism, non-vulnerable
MMIO register design, ...)
order devkits, write test
code to check for vuln
vulnerable ICs
vuln-free ICs

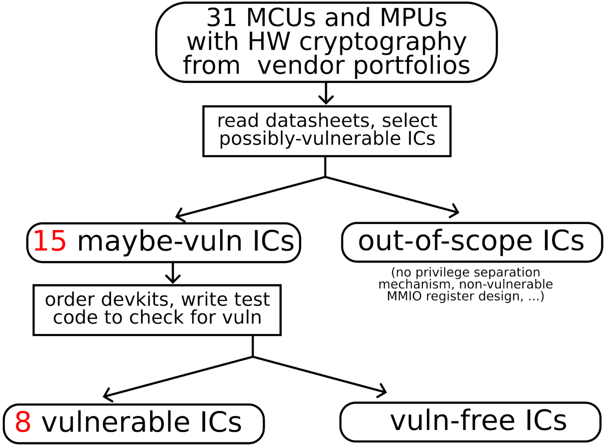
Focus

- Skews towards SoCs with documented accelerators (i.e. micro**controllers**, not micro*processors*)
  - Please publish documentation
  - Attempted reverse-engineering two anyway (Renesas RA2E1, Microchip SAML11)

Focus

- Skews towards SoCs with documented accelerators (i.e. micro**controllers**, not micro*processors*)
  - Please publish documentation
  - Attempted reverse-engineering two anyway (Renesas RA2E1, Microchip SAML11)
- Survey of **SoCs**, not *end-user products*
  - Latter not practical

Results



31 MCUs and MPUs with HW cryptography from vendor portfolios
read datasheets, select possibly-vulnerable ICs
15 maybe-vuln ICs
out-of-scope ICs
(no privilege separation mechanism, non-vulnerable MMIO register design, ...)
order devkits, write test code to check for vuln
8 vulnerable ICs
vuln-free ICs

Results

31 MCUs and MPUs with HW cryptography from vendor portfolios

- 3 different vendors with vulnerable devices
- For every vendor: also have a product **with** countermeasures
- RSA accelerators: hardware bugs??

order devkits, write test code to check for vuln

(no privilege separation mechanism, non-vulnerable MMIO register design, ...)

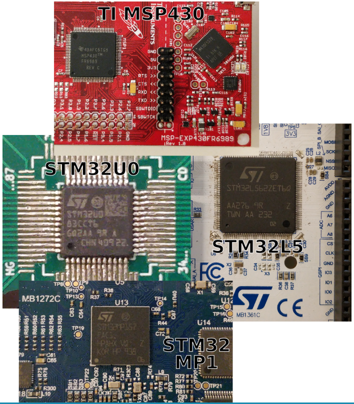
8 vulnerable ICs

vuln-free ICs

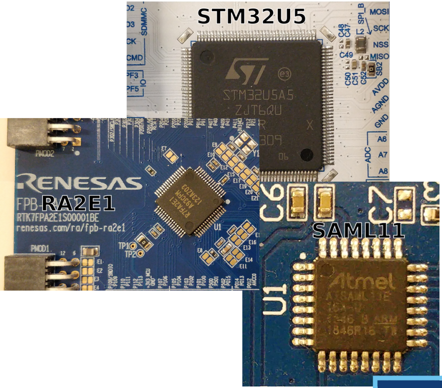
KU LEUVEN

Details

Vulnerable



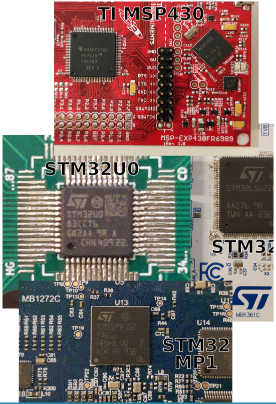
Not vuln.

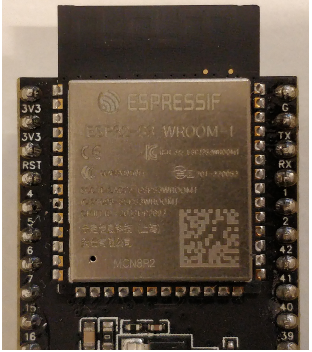


Details

ESP32x3:

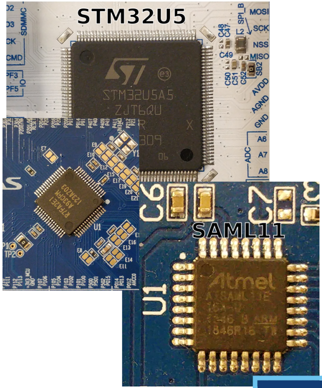
Vulnerable





It's Complicated

Not vuln.



Outline

Conclusion

- 'Simple' but overlooked attack
- Caused not by single faulty component, but complex interaction between components
- Still important to real-world attackers
- Seems to be *slowly* on its way out? (Correlation with introduction of Arm PSA Certified?)

Questions?

Bibliography

[1] Eli Biham and Adi Shamir. “Differential Fault Analysis of Secret Key Cryptosystems”. In: *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*. Ed. by Burton S. Kaliski. CRYPTO ’97. Berlin, Heidelberg: Springer-Verlag, 1997, pp. 513–525. ISBN: 978-3-540-69528-8. URL: https://doc.lagout.org/security/Papers/DFA%20of%20Secret%20Key%20Cryptosystems.pdf.

[2] Márton Bognár, Jo Van Bulck, and Frank Piessens. “Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures”. In: *2022 IEEE Symposium on Security and Privacy (SP)*. 2022, pp. 1638–1655. DOI: 10.1109/SP46214.2022.9833735. URL: https://mici.hu/papers/bognar2022gap.pdf.

[3] Márton Bognár et al. “Intellectual Property Exposure: Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers”. In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 2155–2172. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/bognar.

[4] plutoo, derrek, and smea. *Console Hacking: Breaking the 3DS*. Dec. 27, 2015. URL: https://media.ccc.de/v/32c3-7240-console_hacking (visited on 12/16/2024).

[5] Marc Schink and Johannes Obermaier. “Taking a look into execute-only memory”. In: *Proceedings of the 13th USENIX Conference on Offensive Technologies*. WOOT’19. Santa Clara, CA, USA: USENIX Association, 2019, p. 1. URL: https://www.usenix.org/system/files/woot19-paper_schink.pdf.

[6] Mark Schink and Johannes Obermaier. *Exception(al) Failure - Breaking the STM32F1 Read-Out Protection*. Archived at https://web.archive.org/web/20250318184501/https://blog.zapb.de/stm32f1-exceptional-failure/. Mar. 17, 2020. URL: https://blog.zapb.de/stm32f1-exceptional-failure/ (visited on 04/02/2025).

[7] SciresM and hexkyz. *Je Ne Sais Quoi - Falcons over the Horizon*. Archived at https://archive.is/wNT42. Nov. 19, 2021. URL: https://hexkyz.blogspot.com/2021/11/je-ne-sais-quoi-falcons-over-horizon.html (visited on 12/06/2024).

[8] Samuel Junjie Tan, Sergey Bratus, and Travis Goodspeed. “Interrupt-Oriented Bugdoor Programming: A Minimalist Approach to Bugdooring Embedded Systems Firmware”. In: *Proceedings of the 30th Annual Computer Security Applications Conference*. ACSAC ’14. New Orleans, Louisiana, USA: Association for Computing Machinery, 2014, pp. 116–125. ISBN: 9781450330053. DOI: 10.1145/2664243.2664268.

Other image attributions

- zest (Encryption key icon, MIT)
- Andrew Hussie (Vriska Serket, Sweet Bro & Hella Jeff)
- `mikeazo` on Stack Overflow (AES diagram, CC BY-SA 3.0)
- Apple (WebKit logo, CC BY-SA 4.0)
- ArtyomK2707 (Intel i9-14900K, CC BY-SA 4.0)
- InfoSecDJ (microscopic fuses, CC BY-SA 4.0)
- Lisa Schulz (lock and key icons, CC BY-SA 4.0)