got HW crypto?

On the (in)security of a Self-Encrypting Drive series

Finse Winter School 2018
Gunnar Alendal

Speakers intro

Gunnar Alendal:

Cand.Scient (old skool) in Cryptography from the University of Bergen, UiB, Norway.

Reverse engineering anything with an opcode; x86, x64, ARM, MIPS, M68k, ARC, 8051, ..

Security researcher with 18+ years of professional experience.

Talk motivation

- “Old” research from 2015 (eprint 2015/1002)
- Still very relevant
- Everything is a SoC ⇒ “FW is the new SW”
- **HW/FW** less exposed to security research
- Rarely open source ⇒ Reverse engineering

Research motivation

is HW crypto more secure?

JMS538S X

SW6316 X

OXUF943SE X

INIC-1607E X

JMS569 X

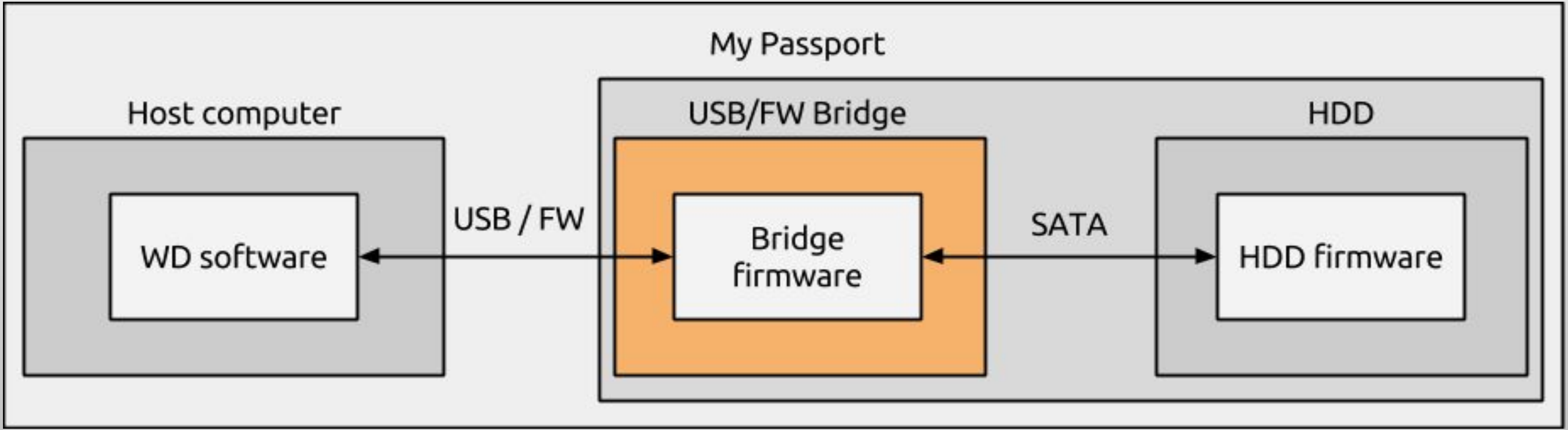
INIC-3608 X

Western Digital My Passport / Book

- Self-encrypting external HDD series*
- Crypto done in either:
  1. 1st-gen : USB/FW-to-SATA bridge
  2. 2nd-gen : HDD itself
- Can’t fit everything in talk ⇒ read full paper

* Some models don’t support encryption

Generic setup



Different USB bridges researched

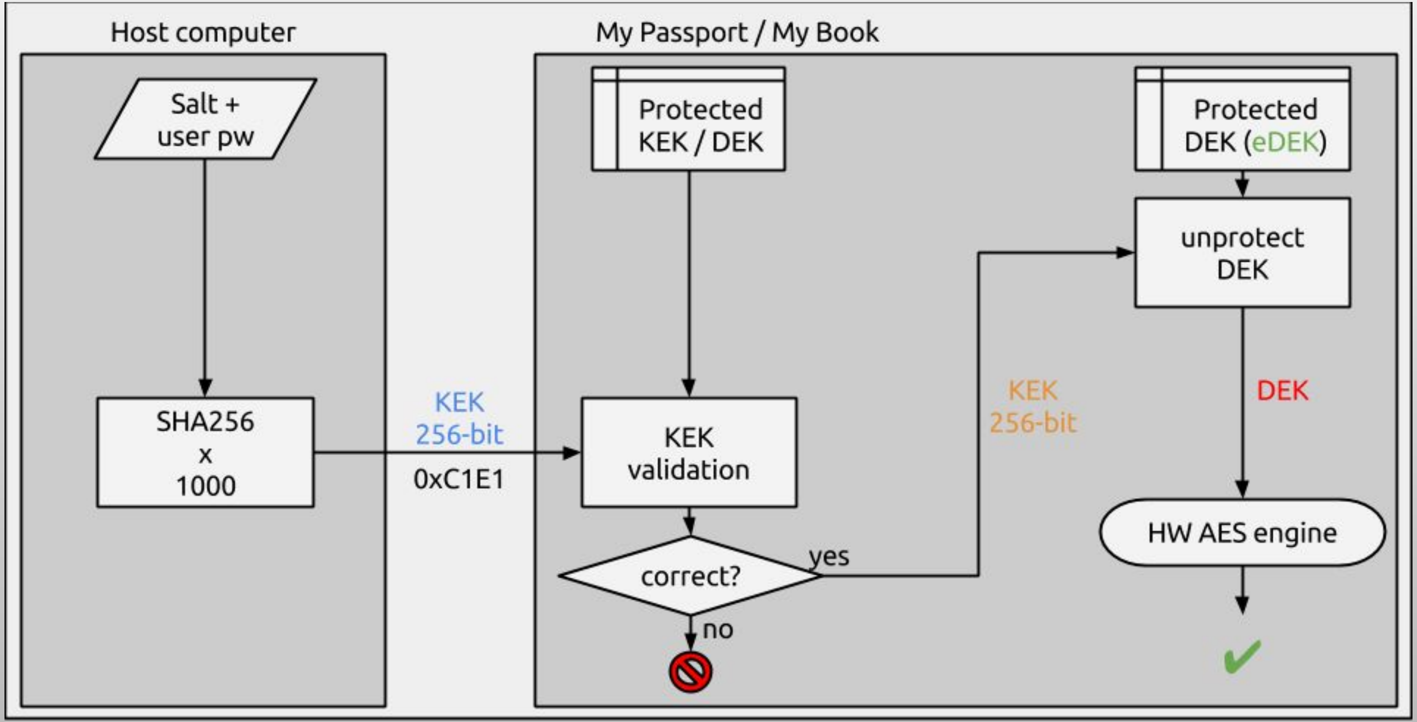
Vendor	Model (1st-gen/2nd-gen)	Architecture
JMicron	JMS538S	Intel 8051
Symwave	SW6316	Motorola M68k
PLX	OXUF943SE	ARM7
Initio	INIC-1607E	Intel 8051
Initio	INIC-3608	ARC 600
JMicron	JMS569	Intel 8051

Overall security design

- User PW ⇒ Key-Encryption-Key (KEK):
  - KDF(salt+PW) = KEK
  - salt + KDF iterations are constant in SW
- KEK protects Data-Encryption-Key (DEK)
- DEK = holy long-term HW AES Key

1st-gen bridges w/AES

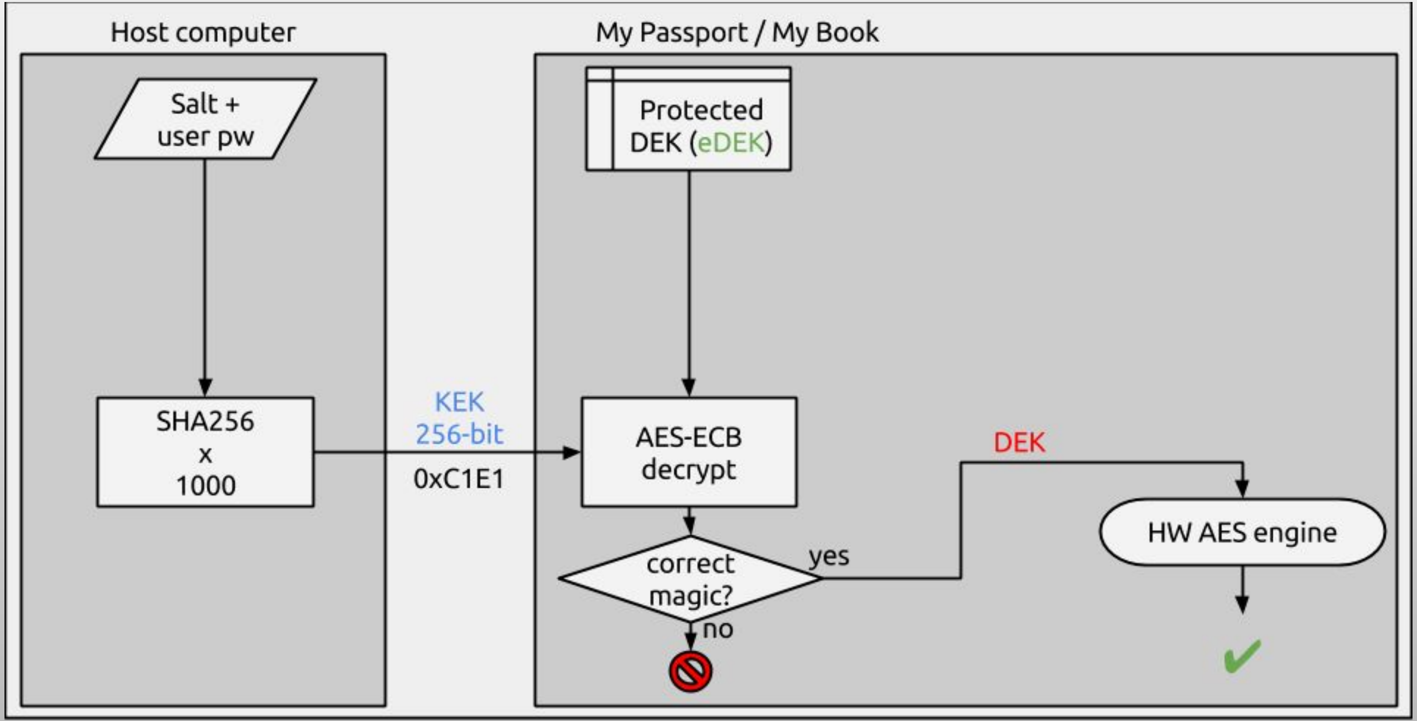
Overall security design



The protected DEK - eDEK

- a KEK-encrypted blob containing the raw DEK
- eDEK stored on disk + USB bridge EEPROM
  - EEPROM is marked “U14” on most PCBs
- retrieve eDEK ⇒ off-device pw brute force

Authentication - JMS538S/INIC-1607E



Mandatory HW encryption

- No PW set ⇔ *hardcoded* KEK unlocks DEK

- Hardcoded KEK = “**PI**” AES-256 key

```
03 14 15 92 65 35 89 79 32 38 46 26 43 38 32 79
FC EB EA 6D 9A CA 76 86 CD C7 B9 D9 BC C7 CD 86
```

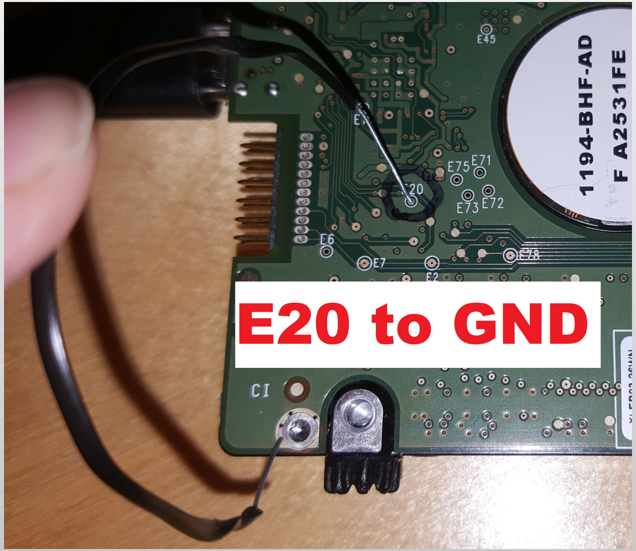
data recovery

- no pw + broken USB bridge? no problem:
  - eDEK stored on HDD + EEPROM
  - decrypt eDEK with “**PI**” KEK ⇒ DEK decrypts HDD

- pw set? off-device brute force
  - Constant salt + KDF iteration counter
  - GPU-impl. benchmark: ~1 mill pw/s (single card)
  - Pre-calculated hash/rainbow-table

Retrieve the eDEK: “no eeprom for you”

- no EEPROM on boot..
- ⇒ raw USB-to-SATA bridge or “DFU mode”
- ⇒ read eDEK from HDD



VID/PID:	1058/0748
Bridge:	JMS538S

Retrieve the eDEK

- JMS538S - “no eeprom for you”
- SW6316 - PC-3k / “no eeprom for you”
- OXUF943SE - SATA + hidden eDEK sector
- INIC-1607E - “no eeprom for you” + 3-byte FW patch to dump eDEK

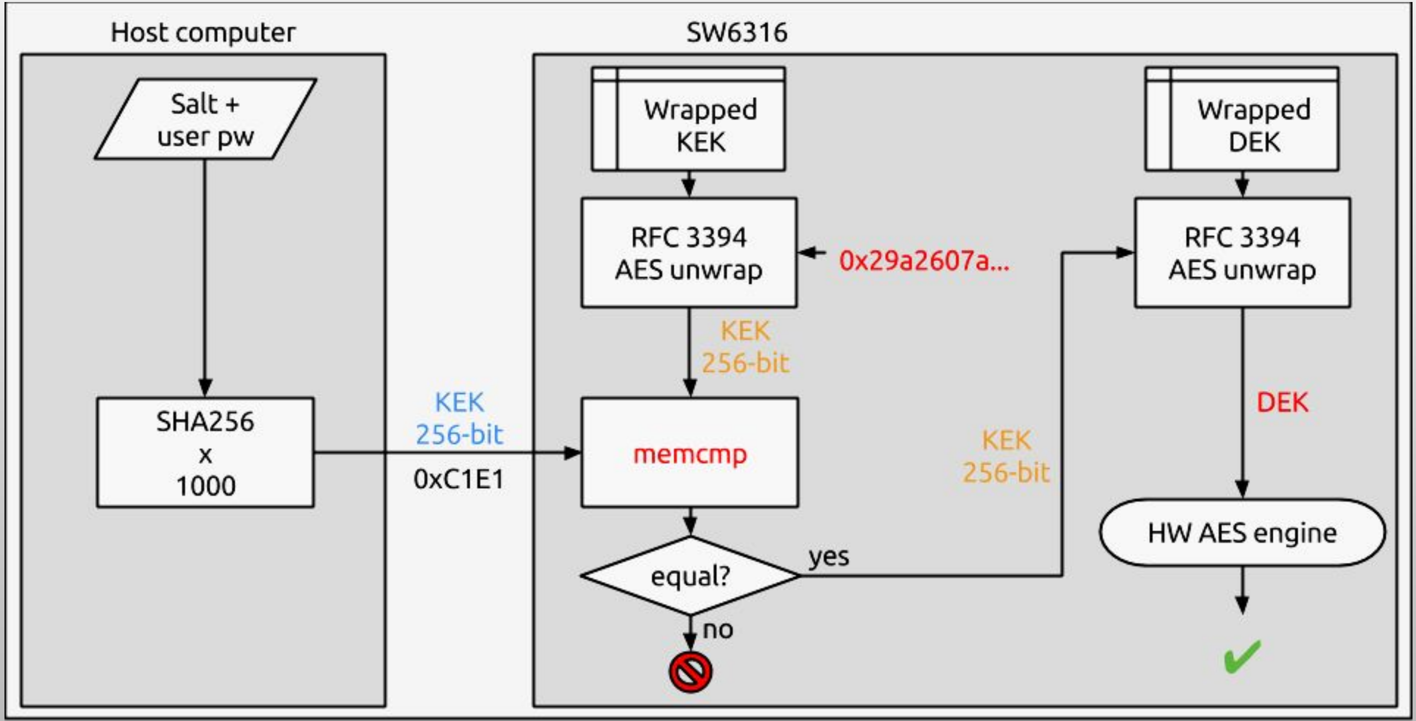
Attackers progress...

Model	no pw set, recovery	pw brute force	break auth.	crack DEK
JMS538S	✓	✓		
SW6316	✓	✓		
OXUF943SE	✓	✓		
INIC-1607E	✓	✓		

Breaking auth. - aka. backdoors

- Two 1st-gen chips fail on authentication
- SW6316 stores the KEK in EEPROM/HDD
  - Protection: Hardcoded key `(0x29A2607A..)`
- OXUF943SE saves a “PI” encrypted eDEK
  - Protection: Hardcoded key `(0x03141592..)`

SW6316 authentication/backdoor



Attackers progress...

Model	no pw set, recovery	pw brute force	break auth.	crack DEK
JMS538S	✓	✓		
SW6316	✓	✓	✓	
OXUF943SE	✓	✓	✓	
INIC-1607E	✓	✓		

..but before we crack DEKs:

2nd-gen bridges with no AES

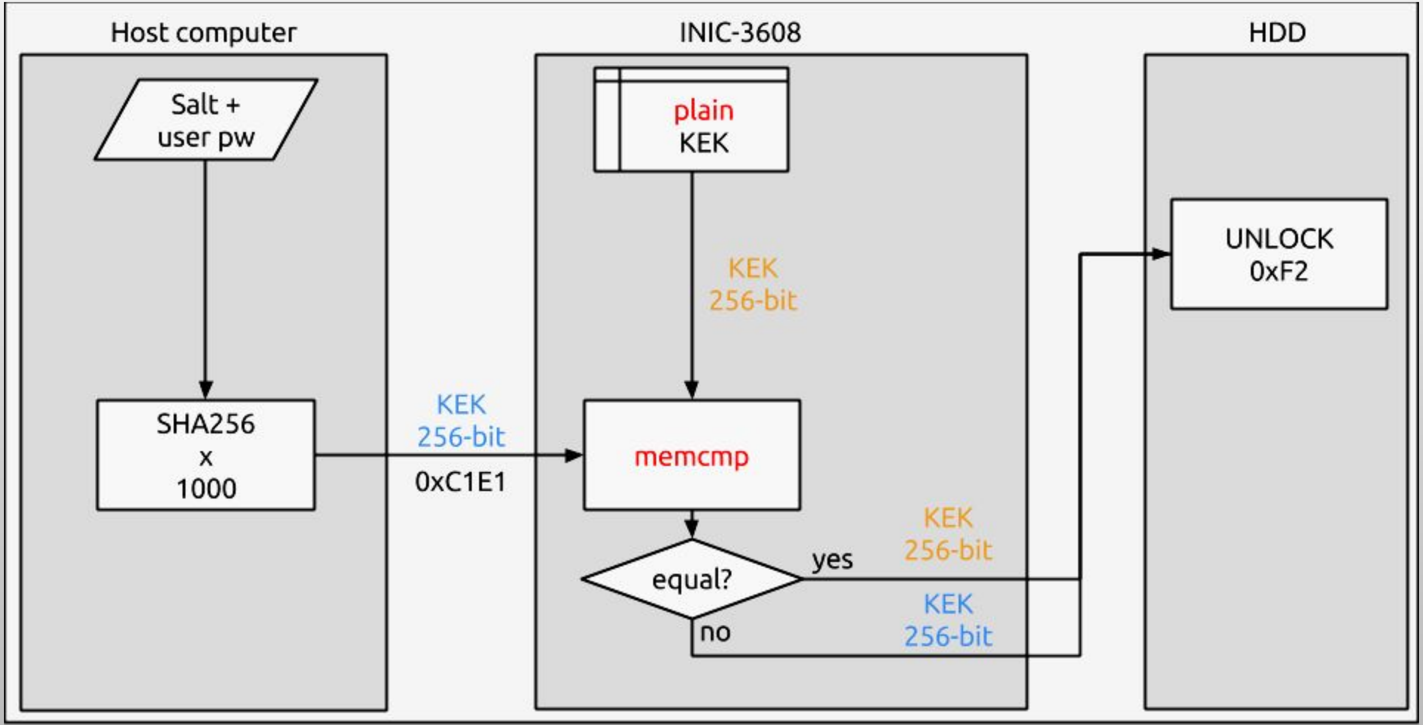
Initio INIC-3608 / JMicron JMS 569

- no HW AES in USB bridge
- HDD does crypto:
    - “ATA Security feature Set”; ATA 0xF1, 0xF2, ...
- VSC “status” (0xC045) reports only cipher mode 0x30 (FDE)

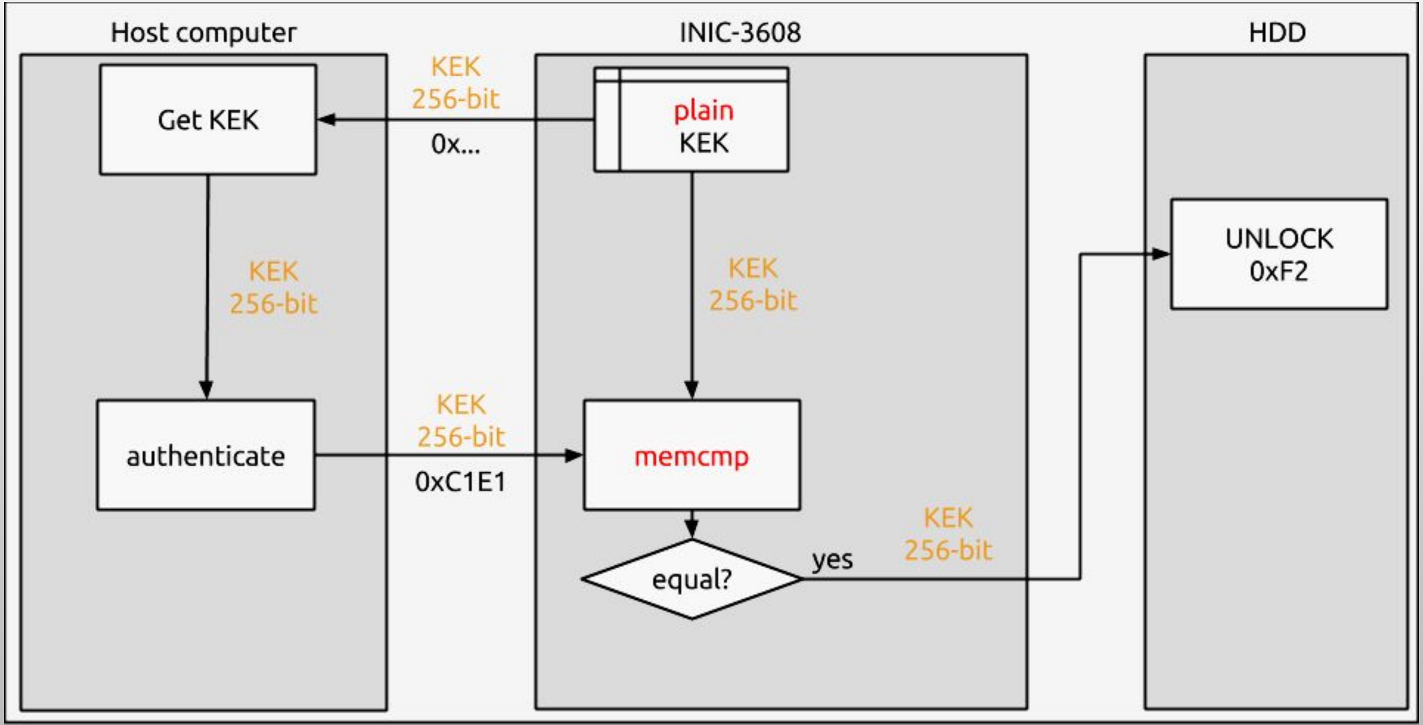
INIC-3608 backdoor

- INIC-3608 does authentication, no crypto
- EEPROM, U14, contains the raw KEK(!)
- Dump EEPROM ⇒ Get KEK ⇒ authenticate
- ..**or** get KEK with secret VSC ⇒ authenticate

INIC-3608 authentication



INIC-3608 backdoor



INIC-3608 Backdoor DEMO

JMicron JMS569

- Connect to pc3k in kernel-mode
  - Get privileges as always by bit shifting
  - Erase ATA-module XX
  - HDD unlocks, decrypting everything on the fly

- By now, pc3k found their own way
  - Details in the forums

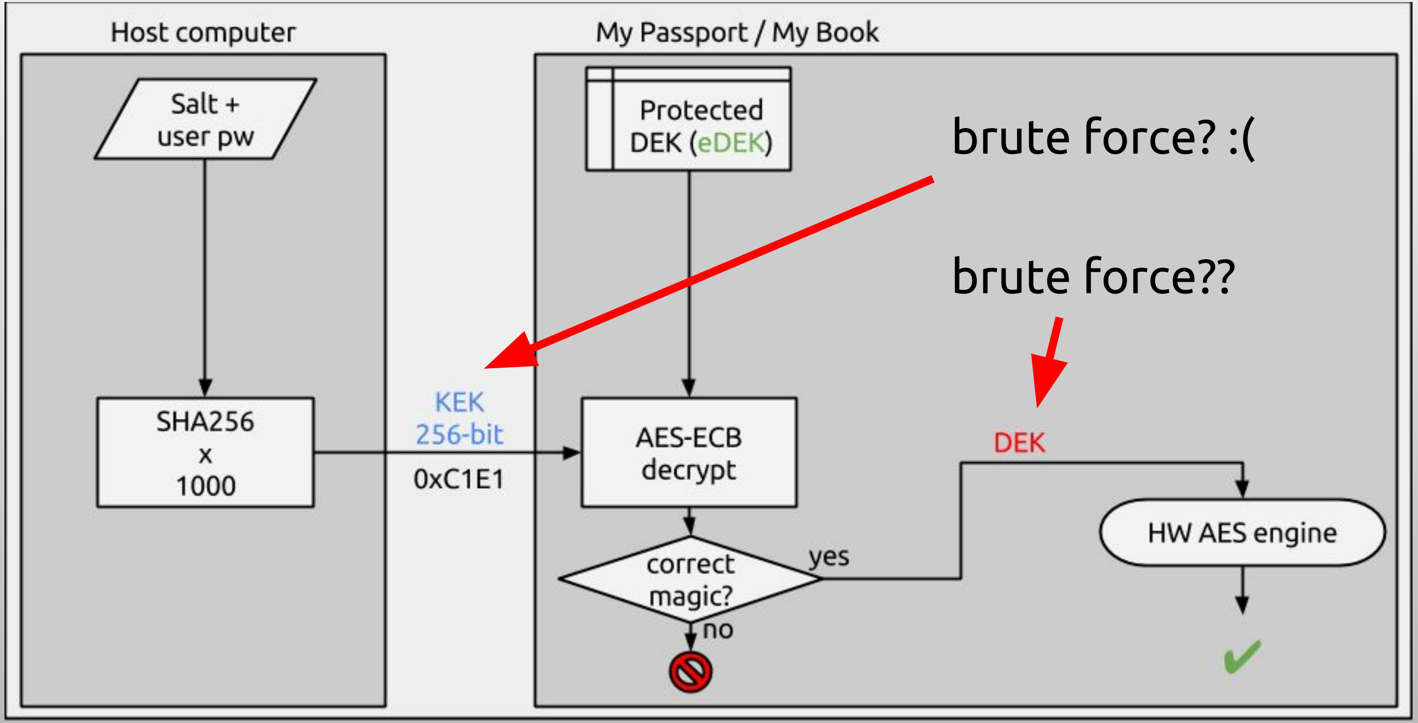
Attackers progress...

Model	no pw set, recovery	pw brute force	break auth.	crack DEK
JMS538S	✓	✓		
SW6316	✓	✓	✓	
OXUF943SE	✓	✓	✓	
INIC-1607E	✓	✓		
INIC-3608	✓	✓	✓	
JMS569	✓		✓	

JMS538S and INIC-1607E still standing tall*

* From the devices available to the researchers

Recap: Authentication - JMS538S



Crack DEK directly?

- How is the HW AES-256 DEK created?

- Entropy source?

- can we beat a 2^{256} complexity?

DEK creation ⇒ device “erase”

- How is the DEK created on a device “erase”?
  - aka. “I forgot my password”

- Entropy source(s)?

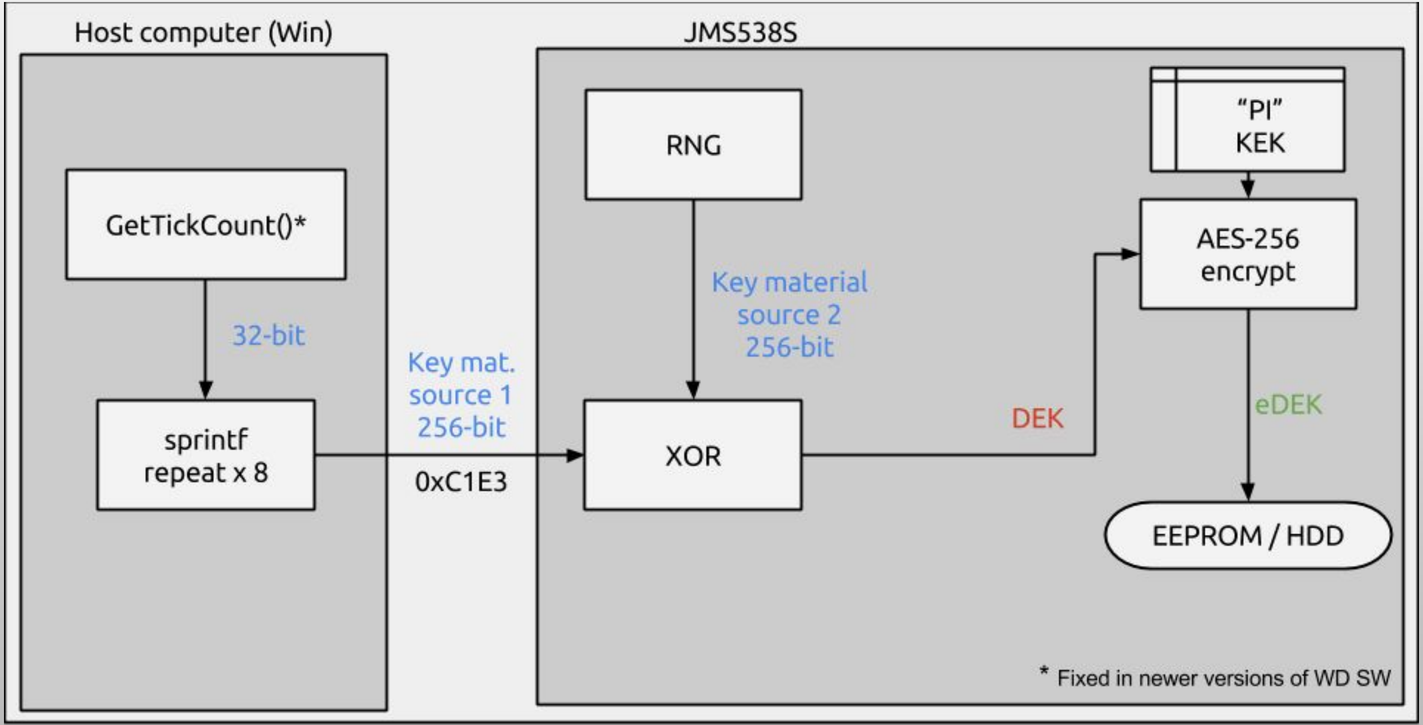
- Can we assume the factory uses this “erase” command?

DEK creation by device “erase”

- “erase” VSC: CDB[0:1] = 0xC1E3

- 2 entropy sources:
  - host computer ⇒ Key material source 1
  - on-board RNG ⇒ Key material source 2

JMS538S “erase” VSC

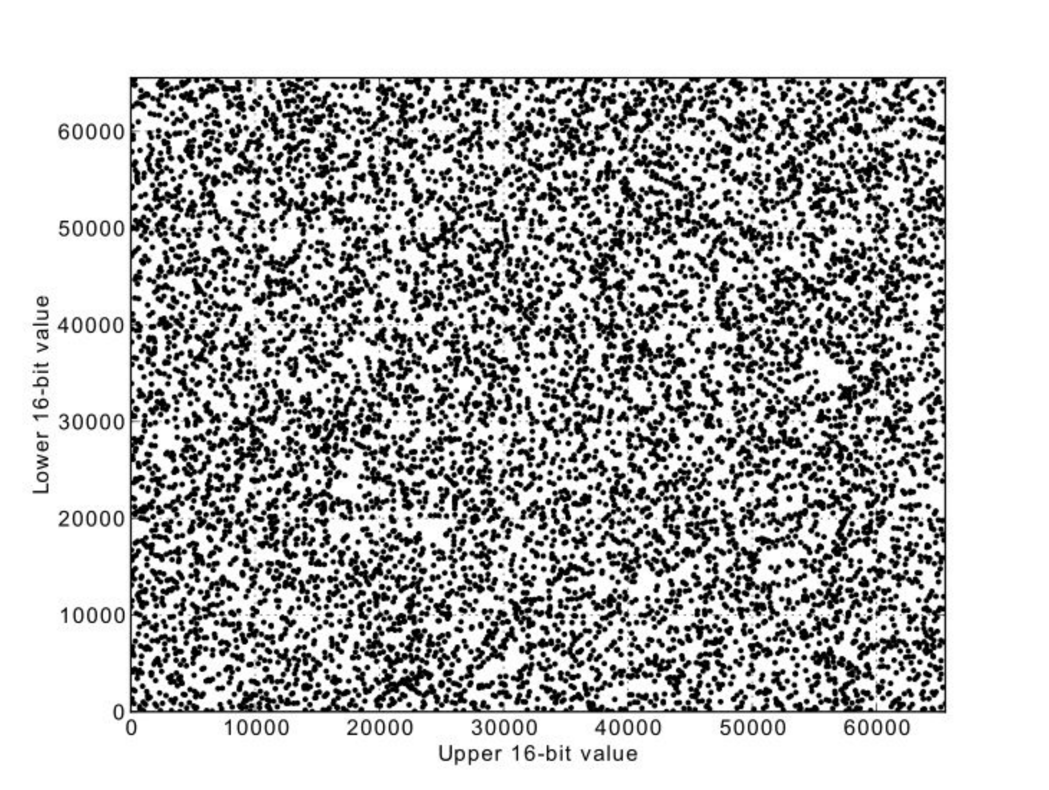


* Fixed in newer versions of WD SW

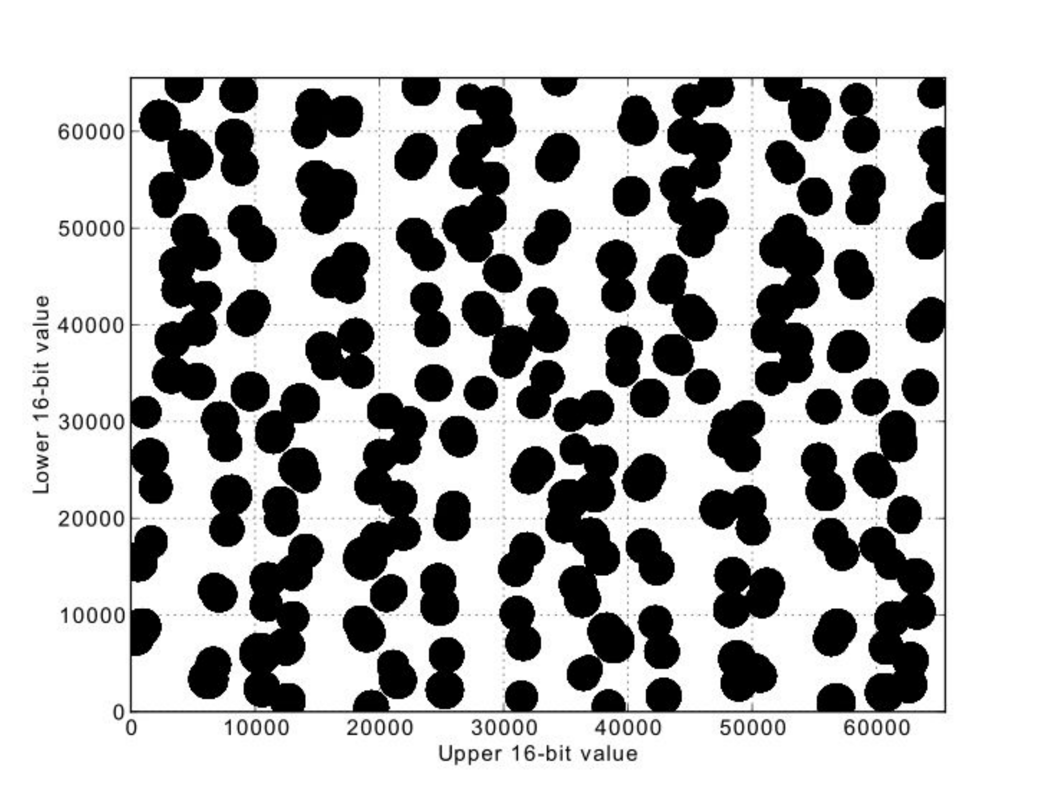
JMS538S on-board RNG

- Implemented in chip “somewhere”
- Gather samples and plot
- Gather by “status” (4 bytes) or “erase” (32 bytes) VSC

/dev/urandom - 32-bit x 10 000



JMS538S “status” unmask x 10 000



JMS538S on-board RNG

- “status” command masks RNG output:
  - xor with 0x271828af
- “erase” uses raw RNG - no mask
- RNG turns out to be a 8-bit LFSR with period 255

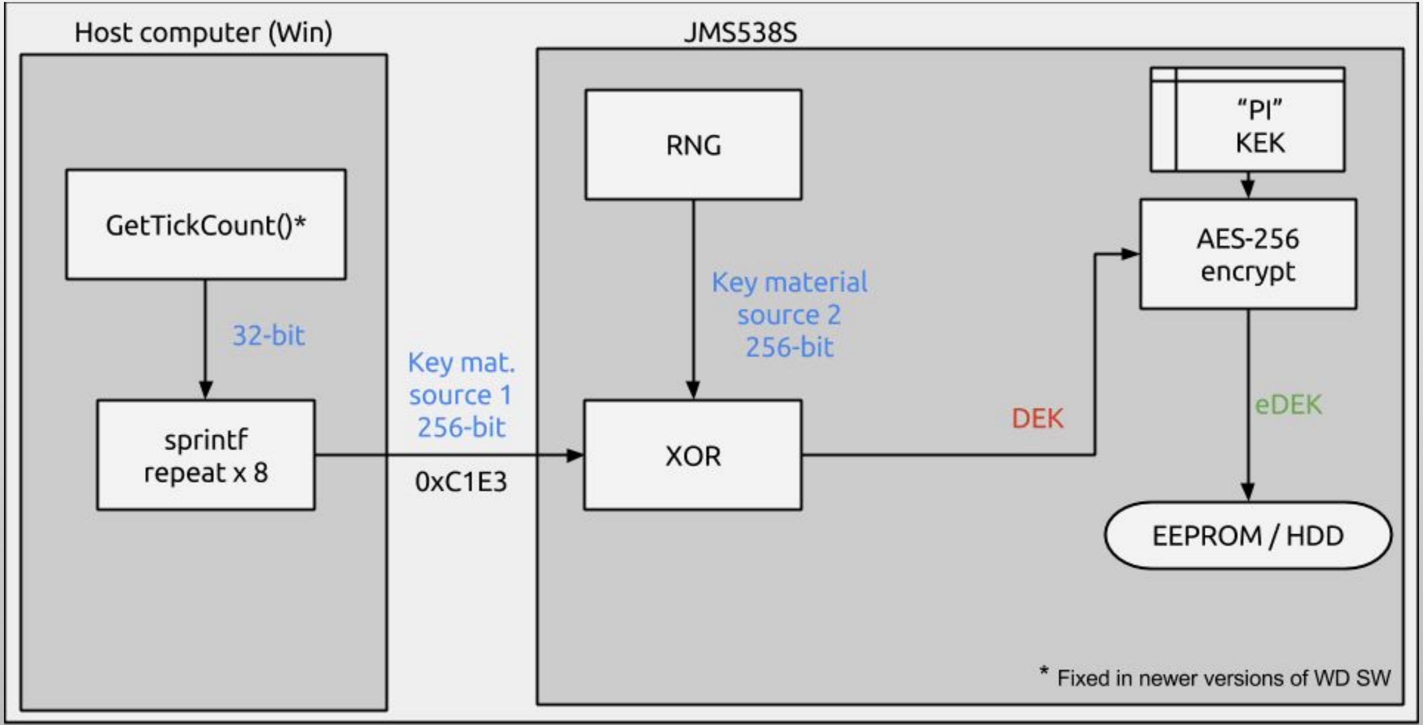
JMS538S on-board RNG

- ..eh, a RNG with period of 255?!
- ..adding a poor ~2^8 to the complexity!
- ..so we have total 2^{32} x ~2^8 = ~2^{40} complexity!

JMS538S “erase” attack

- You erase the drive + set sooper pw
- We recover the DEK with 2^{40} complexity
  - ~2^{36} if set from a MAC
- ..done in “no time” on any computer

JMS538S “erase” VSC



* Fixed in newer versions of WD SW

JMS538S factory keys

- “most people don’t erase their drives”
- ..so what about the factory set DEKs?
- Does the factory use the “erase” command?

JMS538S factory keys analysis

- Grab factory set DEK from an eDEK + reverse the “erase” command flow
- Generate 255 possible “Host provided key material” (source 1)
- Find the correct one by guessing…?

JMS538S factory keys - RNG leak

- The default out-of-the-box eDEK leaks
- Decrypted eDEK leaks RNG status at creation time
- … which is the same time as DEK creation!

decrypted factory eDEK - RNG leak

```
Magic                     0x00: "DEK1"
CRC                       0x04: 3f97
Unknown                   0x06: 0000
random1                   0x08: b1f065be
key 0x3ee2 128 bit        0x0c: dde91629a8f503a41847e9956386a5d3
random2                   0x1c: 2aa98576
key 0x3ef2 128 bit        0x20: fea9c0d0ad395397772420a0563a604b
random3                   0x30: 074195db
key 0x3f02 256 bit        0x34:
3b00e300f7002700e1004d003800040069003e00d70048000c00bb0042006400
random4                   0x54: 8e832cf3
key size (byte)           0x58: 20 => 256 bits
Unknown                   0x59: 00000000000000
```

factory DEK

RNG status leak

JMS538S factory keys - RNG leak

- The default out-of-the-box eDEK says it all
- It gives the raw DEK
- + the *state of the RNG* after DEK creation

- ⇒ We know the host provided key material!

example host provided key material

```
Raw stream: 14 F9 DD 69 49 81 D4 63 CE 22 30 51 23 1B 2C 18 28 3B
3D 15 0F 3F 98 39 E4 C3 1F 4A 57 F3 9A 79


Little endian, 32-bit values: 69DDF914 63D48149 513022CE 182C1B23
153D3B28 39983F0F 4A1FC3E4 799AF357


srand(0x4fd45d3f)       ⇐ Seed with this...
rand() ⇒ 69DDF914         ⇐ ... and get these
rand() ⇒ 63D48149         ⇐ ...
..
rand() ⇒ 799AF357         ⇐ ...
```

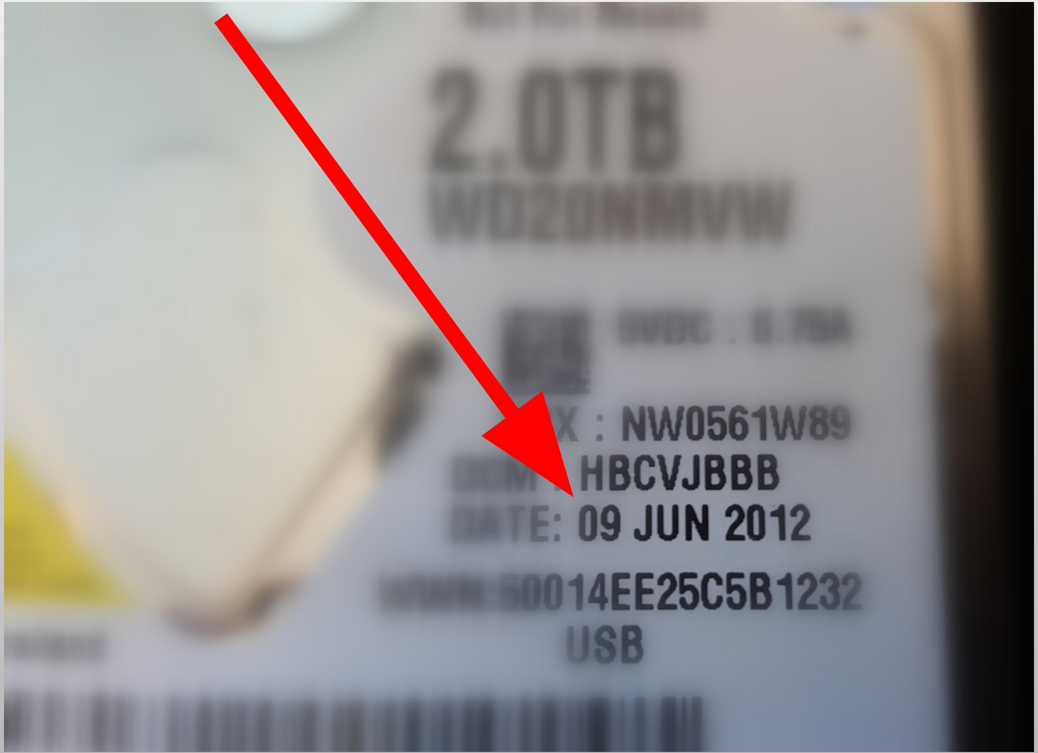
example host provided key material

- srand(0x4fd45d3f) is the entropy source
- 0x4fd45d3f⇒ UNIX time
- 0x4fd45d3f⇒ 2012-06-10 08:39:27 UTC
- It was on a Sunday ..and it was sunny

DEK created: 10 JUN 2012 08:39:27 UTC

Ouch!

HDDs have a printed production date..



JMS538S factory **DEK** attack

- a **single** 128-bit known-plaintext AES block needed from HDD ⇒e.g. $E_{DEK}(00..00)$

- Recover the 256-bit DEK with 2^{36} complexity:
  - Brute force creation time (2007 - 2015) + RNG state

JMS538S factory **DEK** attack

- ..done in “no time” on any computer

- ..or **instant** with a 1.2 TB lookup-table!
  - pre-gen all 2^{36} possible factory DEKs
  - store E_{DEK}(00..00) + seed + RNG idx

JMS538S factory DEK attack DEMO

Attackers progress...

Model	no pw set, recovery	pw brute force	break auth.	crack DEK
JMS538S	✓	✓		✓
SW6316	✓	✓	✓	
OXUF943SE	✓	✓	✓	
INIC-1607E	✓	✓		(✓)
INIC-3608	✓	✓	✓	
JMS569	✓		✓	

badUSB and evil-maid?

No FW signing ⇒ security problems

- can patch FW devices, pre authentication ⇒ bad, bad USB
- ..resulting in spreading of evilness
    - malware in 8051, M68k and ARC. Infect-on-the-fly.
    - no easy clean (self-protecting evil FW)
    - add crypto backdoor
    - nullifying poor auth. schemes

Summary

- **All 6** bridges analyzed had **serious security vulnerabilities**
- **3** bridges have **backdoors**, **2 weak key** setup, **1 broken auth**.
- All 6 vulnerable to unauthorized FW patching ⇒ badUSB, evil-maid, ..

Thank You, WD and EFF

Questions?