OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

A common weakness in RSA signatures: extracting public keys from communications and embedded devices

Hackito Ergo Sum
24-26 April 2014

Renaud Lifchitz
renaud.lifchitz @ oppida.fr

Speaker's bio

- French computer security engineer working at Oppida, France

- Main activities:
  - Penetration testing & security audits
  - Security research
  - Security trainings

- Main interests:
  - Security of protocols (authentication, cryptography, information leakage…)
  - Number theory (integer factorization, primality testing…)

RSA signature basics

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Introduction – Digital signatures

- Asymmetric cryptography is widely used to do digital signatures:
  - Private keys are used to digitally sign messages
  - Corresponding public keys are used to verify signatures
  - Integer fatorization allows an attacker to find the private keys from public ones, but is **generally** hard

- **Public keys are almost always transmitted out-of-band** (public key server, local keystore) before communication/usage

- One of the most used signature scheme is RSA signature

Introduction – RSA signature

- Steps to sign a message using RSA:
  - Message *m* is hashed using a hash algorithm *h( )* : MD5, SHA1, SHA256, …
  - Hash is then padded to avoir forgery by multiplication, using a padding algorithm *p( )* like PKCS
  - The result is raised to the *d*-th power and reduced modulo *n*, where *d* is the private exponent and *n* is the public key

$$p(h(m))^d \equiv s \pmod{n}$$

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Extracting public keys from signed messages

The idea

- Suppose we have 2 different messages with their corresponding signatures *(m_1,s_1)*, *(m_2,s_2)* with unknown public key *n*:

$$\begin{cases} p(h(m_1))^d \equiv s_1 \pmod{n} \\ p(h(m_2))^d \equiv s_2 \pmod{n} \end{cases}$$

$$\Rightarrow \begin{cases} p\big(h(m_1)\big) \equiv {s_1}^e \pmod{n} & \text{with quotient } q_1 \\ p(h(m_2)) \equiv {s_2}^e \pmod{n} & \text{with quotient } q_2 \end{cases}$$

by Euler theorem

$$\Rightarrow \gcd\big({s_1}^e - p\big(h(m_1)\big), {s_2}^e - p(h(m_2))\big) = \gcd(q_1, q_2) \,.\, n$$

which gives a small (probably smooth) multiple of public key *n*

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

The idea

- Then we have to remove all small factors from the result until the residue size is a well-known asymmetric key size (512, 768, 1024, 2048, 4096 bits…)

- Trial division is sufficient in 99,9999 % of cases, otherwise we can use an additional signed message in the GCD or use ECM factoring algorithm to help

- We now have computationally extracted our unknown public key!

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Requirements

- Hash and padding algorithms must be known or guessed
- *e* should be small because computation will be done without modular arithmetic
- *n* should be small to medium

Complexity

- Main limitation is memory consumption

- The computation:
  - takes about $O(e.\log(n))$ bits of memory
  - costs about:
    - $O(\log(e))$ big integer multiplications (exponentiation step)
    - $O(e.\log(n))$ big integer divisions (GCD step)

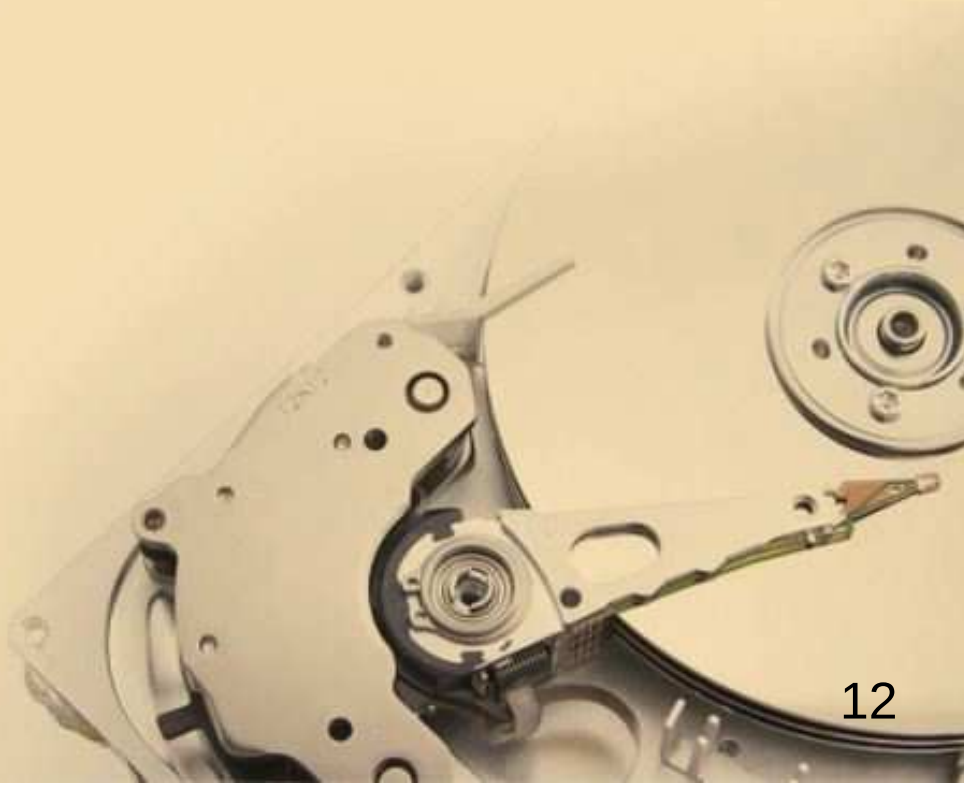
OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Applications

- Without access to any kind of keyserver nor keystore and being entirely passive, we can:
  - Extract public keys used in RSA signatures
  - Authenticate subsequent messages
  - Find people or devices using weak keys that weren't discoverable before: **this gives a new angle of attack for embedded devices/blackbox protocols using RSA signatures**
  - Safely test whether different messages are signed using the same key/come from the same person
    (without relying on any kind of spoofable key id)

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

State of the art of factorization algorithms



Introduction

- There exists several algorithms for integer factorization, more or less naive
- Some algorithms are generic and can factor any number, some are form-specific
- Key generation weaknesses:
  - p and q too close
  - p-1, q-1, p+1 and/or q+1 too smooth
  - weak RNG (Random Number Generator)
- A generic but good open source program for factoring: Yafu (http://sourceforge.net/projects/yafu/)

Finding small factors in large integers

- Trial factoring:
  when there are very small factors (less than 10 digits)
- Pollard Rho:
  for small factors
- Pollard's P-1:
  when one or more factors are p-1 smooth
- Williams' P+1:
  when one or more factors are p+1 smooth
- Elliptic Curve Method (ECM):
  for factors up to 80 digits

Finding large factors in small integers

- Fermat algorithm:
  when a factor and its co-factor are really near in absolute value
- Quadratic sieve (QS):
  faster and simpler NFS for integers < 100 digits
- Number Field Sieve (NFS):
  for integers of intermediate size
- General Number Field Sieve (GNFS):
  for numbers up to 230 digits (RSA-768)
- Special Number Field Sieve (SNFS):
  for numbers with specific form ($r^e \pm s$ with *r* and *s* small)
  up to 320 digits

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Practical applications - PGP

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

What is PGP?

- Pretty Good Privacy (PGP) is a data encryption and decryption program mostly used for securing e-mails
- Created in 1991 by Phil Zimmermann
- Software: PGP (Windows) / GnuPG (Linux)
- OpenPGP standard (RFC 4880)

Computation steps to extract public key - PGP

- Prepare original message before hashing:
  - Canonicalize message (newlines are converted to `\r\n`)
  - Append specific PGP data:
    - PGP version
    - Signature type
    - Public algorithm (here RSA)
    - Hash algorithm
    - Signature date & time
- Recreate PKCS#1 padded ASN.1 message hash following RFC 4880
- Compute:

$$\gcd\left(s_1{}^{65537} - p\left(h(m'_1)\right),\ s_2{}^{65537} - p\left(h(m'_2)\right)\right)$$

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Proof-of-concept implementation

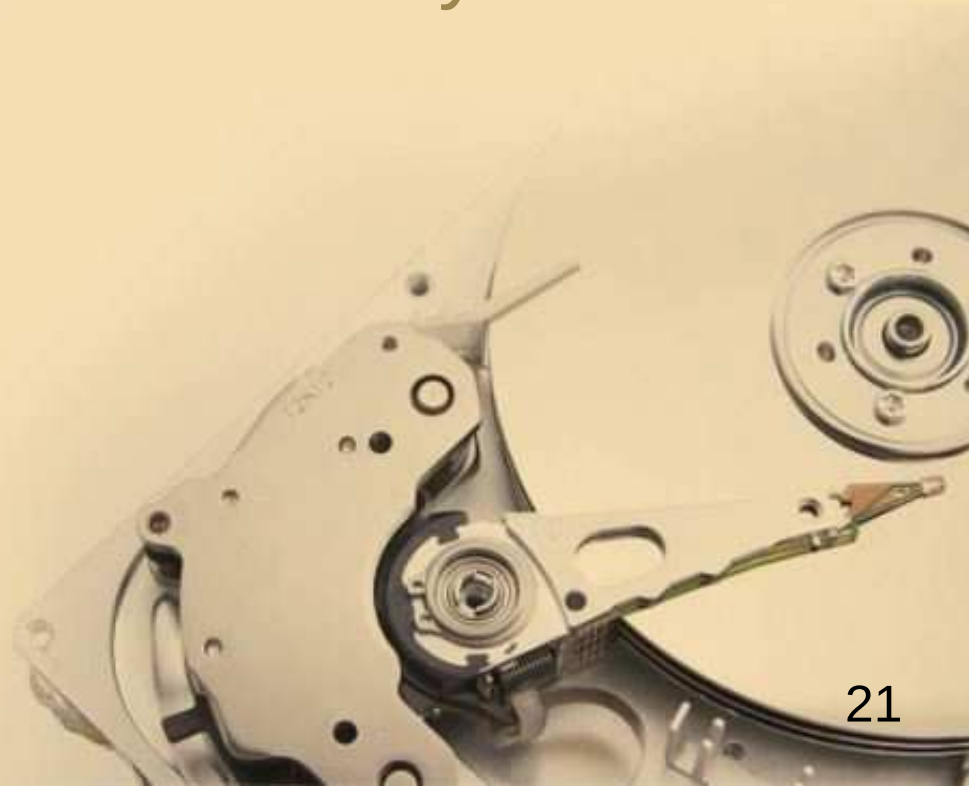
- Just a proof-of-concept:
  - Supports RSA signature with SHA-1 hashing only
  - Not optimized (mixed Python + PARI-GP implementation, would be faster in C)
- Able to find the signing public key of anybody using only 2 signed mails!

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Proof-of-concept implementation

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

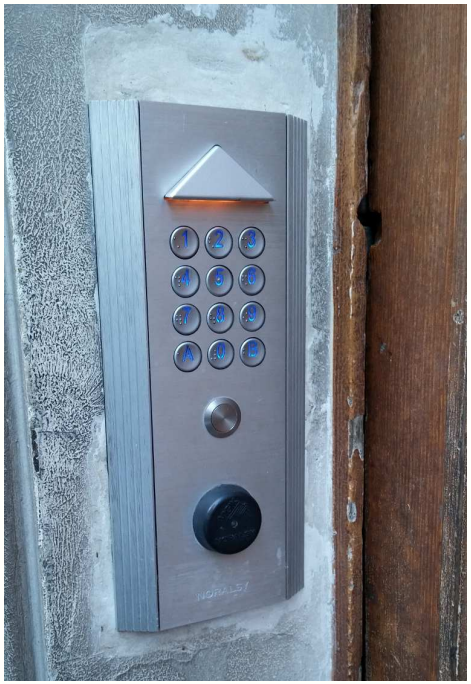
Practical applications - Vigik access control system



What is Vigik?

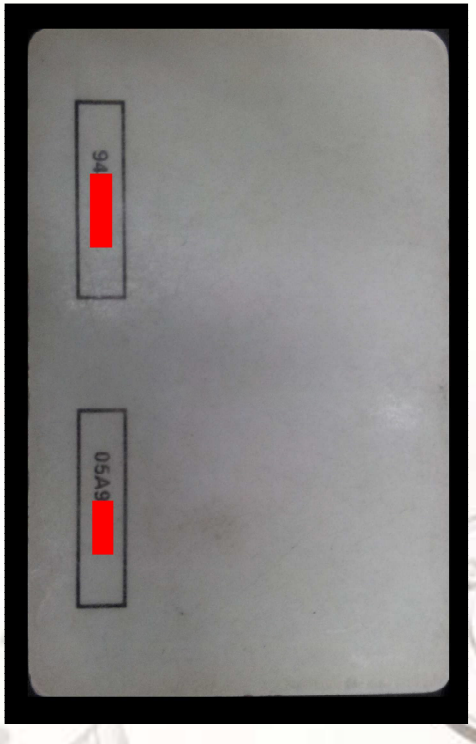
- French access control for residential buildings (nearly 1 million buildings are protected by Vigik in France)
- Contactless system
- Made to replace the old T25 lock and avoid existing master keys
- 2 kinds of tokens:
  - Resident tokens (various contactless protocols, not interesting), can access a given building at any time
  - Service tokens (based on Mifare Classic + RSA signature of 768 or 1024 bits), can access all buildings during specific time slots
- May be used for other kinds of access control like ATMs or military premises

What is Vigik?



Vigik contactless reader

Resident token



Service token

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

What is Vigik?

- 4 common types of service tokens:
  - « La Poste Service Universel »:
    service code 0x7AA, authorized access from Monday to Saturday, 6:00-0:00 (may vary)
  - « La Poste Autre Services »:
    service code 0x7AB, authorized access any day, 6:00-0:00 (may vary)
  - « France Telecom »:
    service code 0x7AC, authorized access any day, any time
  - « EDF-GDF » :
    service code 0x7AD, authorized access any day, any time

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

What is Vigik?

- Service tokens need to be loaded with a valid RSA signature for the current date & time slot
- For instance, the postmen load their token every morning before mail delivery
- A token can be loaded in advance but for no more than 3 slots of 84 successive hours for security reasons (to mitigate token loss or theft risks)

Vigik storage

- Vigik uses NXP Mifare Classic 1K cards as storage
- 16 sectors of 4 blocks = 64 blocks of 16 bytes
- Last block of each sector is reserved for A and B keys and ACL
- RSA signatures are splitted across several blocks/sectors

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Attacks against protocol
Reader-only attacks

- According to NXP, manufacturer of Mifare products, Vigik has registered prefix code 0x49 (see NXP AN10787 document)
- With no valid service token, using a blank Mifare Classic card, and by crafting several MAD (Mifare Application Directory) structures, sniffing using a Proxmark3 RFID device, we have noticed that a 0x4910 entry triggers a sector 1 read
- Sector authentication in Mifare Classic is badly designed: reader authenticates itself first. It is possible for the card to send many challenges and gather all the answers for an offline cracking
- Using a crapto1 library, it becomes possible to crack the 48-bit sector access key A which happens to be:

0x314B49474956

(« 1KIGIV » in ASCII, to be read in reverse order…)

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Attacks against protocol
Card-only attacks

- With no valid Vigik reader, it is possible to retrieve all sector keys using the well-known offline nested attack by Nethemba (`mfoc` tool), as sector 0 key A is default key 0xA5A4A3A2A1A0

- We find that key A for other sectors is « 1KIGIV » and that key B is proprietary and can vary between cards

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Attacks against protocol

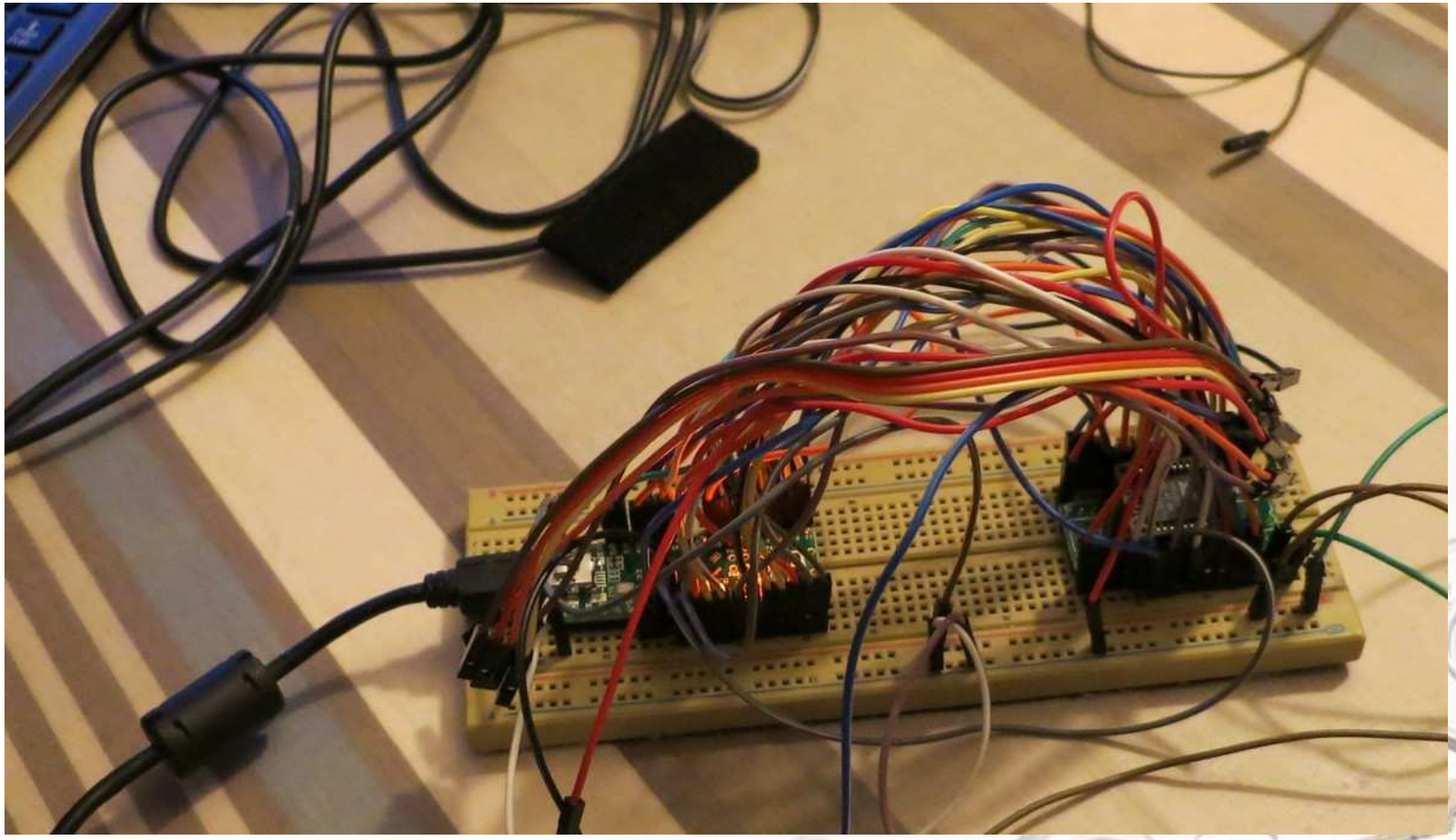
- With the knowledge of A and B keys, we are now able to:
  - Dump and analyze any service token
  - Clone any service token
    (for instance using a Chinese programmable UID Mifare Card)
  - Emulate any service token
    (for instance using a Proxmark3)

Vigik card layout reverse-engineered

Vigik	
id	0x[redacted]
id_checksum	0x[redacted]
manufacturer_data	'\x08\x04\x00bcdef[...]
MAD_checksum	0x44
MAD_null	0x0
Vigik_sector_id1	0x4910
Vigik_sector_id2	0x4916
Vigik_sector_id3	0x4916
Vigik_sector_id4	0x4916
additional_sector_ids	''
s0_keyA	'\xa0\xa1\xa2\xa3\[...]
s0_ACL	'ig\x89\xc1'
s0_keyB	'VLP_MA'
counters	'\x05'
Vigik_service_code	0x7aa
Vigik_info_flags	33L
Vigik_key_version	2
ptr_counter	0x8
counter_num	0x0
slot_access_date	2012/04/20 06:00
slot_dst_duration	1080
other_slots	''
services_counter	1
loading_date	2012/04/20 08:09
reserved_null	0
s1_keyA	'1KIGIV'
s1_ACL	0x79678800
s1_keyB	'\xba[\x89]\xa1b'
RSA_sig_1	'/\xfd\xe4\x88\xfd[...]
s2_keyA	'1KIGIV'
s2_ACL	0x78778800
s2_keyB	'\xba[\x89]\xa1b'
RSA_sig_2	'~\xe5(p\xa4\xc6[\[...]
s3_keyA	'1KIGIV'
s3_ACL	0x78778800
s3_keyB	'\xba[\x89]\xa1b'
RSA_sig_3	'\xbd\xc2.\xdb\xa3[...]
s4_keyA	'1KIGIV'
s4_ACL	0x78778800
s4_keyB	'\xba[\x89]\xa1b'

```
[redacted] [redacted] 08 04 00 62 63 64 65 66 67 68 69
44 00 10 49 16 49 16 49 16 49 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0 a1 a2 a3 a4 a5 69 67 89 c1 56 4c 50 5f 4d 41
05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
aa 07 00 00 21 02 08 00 00 70 04 14 06 00 38 84
00 00 00 00 00 00 00 00 01 70 04 14 08 09 00 00
31 4b 49 47 49 56 79 67 88 00 ba 5b 89 5d a1 62
2f fd e4 88 fd 73 b9 e5 d5 6e c9 28 88 3e df 94
da b8 79 fc 19 b9 65 b4 8a 46 de 01 23 3f a0 fe
d2 53 54 ed c5 16 46 35 92 8b c4 fe a9 43 e1 34
31 4b 49 47 49 56 78 77 88 00 ba 5b 89 5d a1 62
7e e5 28 70 a4 c6 5b 97 c5 39 26 45 4e b9 91 e8
40 1a 37 0f 80 03 f6 43 f5 1f eb 14 43 3d 18 72
54 73 3b 17 1f e9 4c 0c f7 12 f9 98 28 c2 41 00
31 4b 49 47 49 56 78 77 88 00 ba 5b 89 5d a1 62
bd c2 2e db a3 e5 67 55 0a d9 04 26 8c 65 6a 11
63 eb c1 21 49 3f 62 a5 cf 67 0a 93 73 c1 cc 01
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
31 4b 49 47 49 56 78 77 88 00 ba 5b 89 5d a1 62
```

Hardware attack: dumping the reader flash memory



Dumping the flash memory of a Vigik reader using a Teensy 2
(thanks to Gric for his help!)

Hardware attack: dumping the reader flash memory

- Extracted 1024-bit public keys (1/2):
  - « La Poste Service Universel »:
    0xAB9953CBFCCD9375B6C028ADBAB7584BED15B9CA037FADED976599
    6F9EA1AB983F3041C90DA3A198804FF90D5D872A96A4988F91F2243B
    821E01C5021E3ED4E1BA83B7CFECAB0E766D8563164DE0B2412AE4E6
    EA63804DF5C19C7AA78DC14F608294D732D7C8C67A88C6F84C0F2E3F
    AFAE34084349E11AB5953AC68729D07715
  - « La Poste Autres Services »:
    0xA6D99B8D902893B04F3F8DE56CB6BF24338FEE897C1BCE6DFD4EBD
    05B7B1A07FD2EB564BB4F7D35DBFE0A42966C2C137AD156E3DAB6290
    4592BCA20C0BC7B8B1E261EF82D53F52D203843566305A49A22062DE
    CC38C2FE3864CAD08E79219487651E2F79F1C9392B48CAFE1BFFAFF4
    802AE451E7A283E55A4026AD1E82DF1A15

Hardware attack: dumping the reader flash memory

- Extracted 1024-bit public keys (2/2):
  - « France Telecom »:
    0xC44DBCD92F9DCF42F4902A87335DBB35D2FF530CDB09814CFA1F4B
    95A1BD018D099BC6AB69F667B4922AE1ED826E72951AA3E0EAAA7D49
    A695F04F8CDAAE2D18D10D25BD529CBB05ABF070DC7C041EC35C2BA7
    F58CC4C349983CC6E11A5CBE828FB8ECBC26F08E1094A6B44C8953C8
    E1BAFD214DF3E69F430A98CCC75C03669D
  - « EDF-GDF »:
    0xB35193DBD2F88A21CDCFFF4BF84F7FC036A991A363DCB3E802407A
    5E5879DC2127EECFC520779E79E911394882482C87D09A88B0711CBC
    2973B77FFDAE40EA0001F595072708C558B484AB89D02BCBCB971FF1
    B80371C0BE30CB13661078078BB68EBCCA524B9DD55EBF7D47D9355A
    FC95511350CC1103A5DEE847868848B235

Vigik RSA signature

- I have discovered that Vigik uses deprecated ISO 9796-2 for RSA signature with:
  - Public key $N \equiv 5 \pmod 8$
  - p and q of the form 8k+3 and 8k+7 (without order)
  - Public exponent for speed purposes is $e = 2$ (even)

    and $e.d \equiv 1 \left(mod\ \frac{(p-1).(q-1)}{4}\right)$
  - It implies $d = \frac{(p-1).(q-1)+4}{8}$
- It follows that Vigik is vulnerable to some attacks described in:

  « Cryptanalysis of ISO/IEC 9796-1 » by D. Coppersmith, J.S. Coron, F. Grieu, S. Halevi, C. Jutla, D. Naccache, and J.P. Stern

  but chosen-plaintext attacks are not possible in this case

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Vigik security in the next few years

- Interestingly, RSA key for « La Poste Service Universel » has already been changed (key version = 2 in the dump), has the key been compromised?
- Token storage (Mifare Classic) is broken since several years now
- Token signature is within range of direct factoring attacks because weak public keys can be extracted:
  - RSA 768 is already broken (December 2009)
  - RSA 1024 will probably be publicly broken by researchers within 3-4 years

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Vigik security in the next few years

- Because of hardware and storage constraints, key sizes in Vigik are not upgradable (maximum keysize is 1024 bits)
- Full service token forgery will happen in the next few years
- Vigik system is to be changed
- Replacement of 1 million Vigik readers will cost several hundred million euros, upgradable security would have saved this cost

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Countermeasures

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Countermeasures

- The « problem » comes from deterministic padding. RSA encryption uses random padding to avoir various attacks. This is not the case in RSA signature. It would be possible to use non-deterministic padding in signature to avoid public key leaks (like RSA-PSS scheme)

- Other signatures schemes may or may not be vulnerable to this attack (this exercise is left to the reader!)

- In all cases, use strong keys and large enough public exponents

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Thanks!

Any questions?