Bridging Theory and Practice in Cryptography

Pascal Junod

HEIG-VD

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

How large is the gap between academic and industrial-practical-implemented cryptography ?

Outline

- The good
- The bad
- The ugly

The Good

SANS TOP25

Software Error Category: Porous Defenses

[5] CWE-285: Improper Access Control (Authorization)
If you don't ensure that your software's users are only doing what they're allowed to, then attackers will try to exploit your improper authorization and...MORE >>

[6] CWE-807: Reliance on Untrusted Inputs in a Security Decision
Driver's licenses may require close scrutiny to identify fake licenses, or to determine if a person is using someone else's license. Software developers...MORE >>

[10] CWE-311: Missing Encryption of Sensitive Data
If your software sends sensitive information across a network, such as private data or authentication credentials, that information...MORE >>

[11] CWE-798: Use of Hard-coded Credentials
Most of the CWE Top 25 can be explained away as an honest mistake; for this issue, though, customers...MORE >>

[19] CWE-306: Missing Authentication for Critical Function
In countless action movies, the villain breaks into a high-security building by crawling through heating ducts...MORE >>

[22] CWE-732: Incorrect Permission Assignment for Critical Resource
If you have critical programs, data stores, or configuration files with permissions that make your resources accessible to the world - well, that's just what they'll become...MORE >>

[24] CWE-327: Use of a Broken or Risky Cryptographic Algorithm
You may be tempted to develop your own encryption scheme in the hopes of making it difficult for attackers to crack. This kind of grow-your-own cryptography is a welcome sight to attackers...MORE >>

Source: http://www.sans.org/top25-software-errors

Right Key Length

- Estimating the right key length has been one of the first tasks of early cryptography !
- As of today, every crypto student/engineer knows that 512 bits is too short for RSA and too large for a block cipher.

Right Key Length ?

- TI calculators secure boot
- TLS ugly cipher suites
- DVB Common Scrambling Algorithm

TI-x Secure Boot & RSA

TI-83 Plus OS Signing Key Cracked

Posted by Michael on 31 July 2009, 15:33 GMT

The ever-mysterious Benjamin Moody posted a cryptic message on the United-TI forum yesterday. In it, he listed the factorization of the 512-bit RSA modulus used by TI's OS signing key for the 83+ (the "0004 key"). No other details are yet available about how he achieved this feat of substantial brute forcing power. In the event of United-TI downtime, Brandon Wilson has put a copy of Benjamin's values on his personal website.

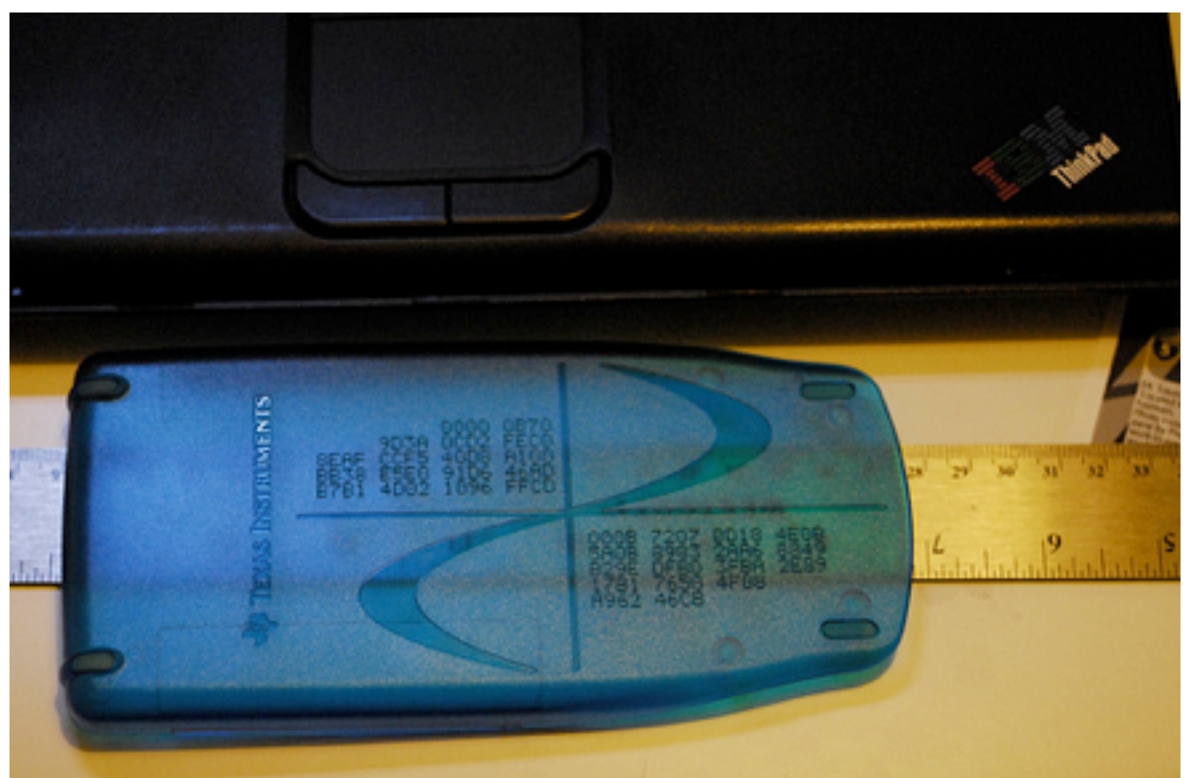
With this achievement, any operating system can be cryptographically signed in a manner identical to that of the original TI-OS. Third party operating systems can thus be loaded on any 83+ calculators without the use of any extra software (that was mentioned in recent news) Complete programming freedom has finally been achieved on the TI-83 Plus!

Update: Benjamin has posted additional details on the United-TI forum thread.

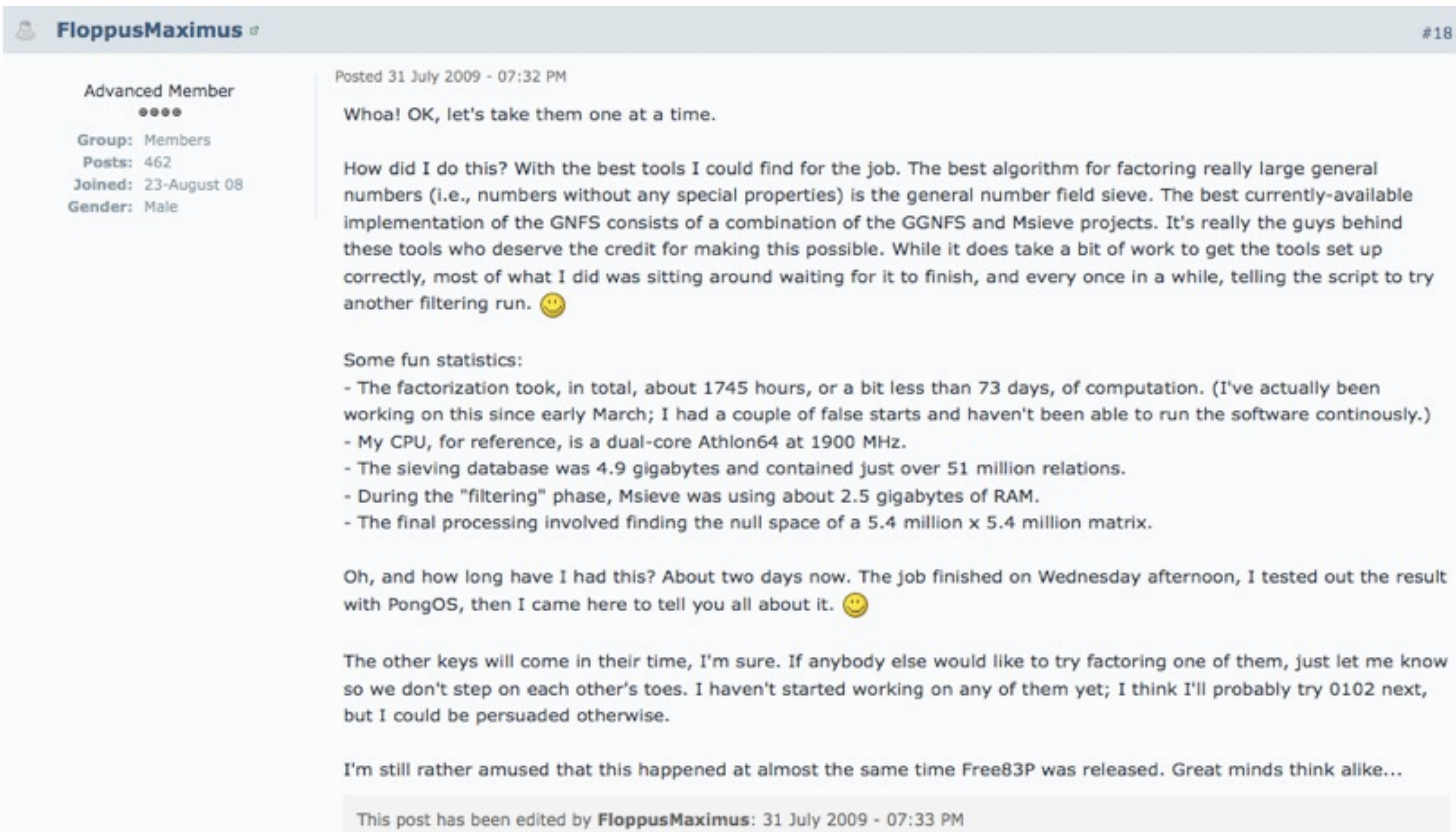
Update: A distributed computing project has been set up. Information about how to join the effort to crack the OS keys for the remaining TI models can be found here.

Reply to this article

TI-x Secure Boot & RSA



TI-x Secure Boot & RSA



FloppusMaximus #18

Advanced Member

Group: Members
Posts: 462
Joined: 23-August 08
Gender: Male

Posted 31 July 2009 - 07:32 PM

Whoa! OK, let's take them one at a time.

How did I do this? With the best tools I could find for the job. The best algorithm for factoring really large general numbers (i.e., numbers without any special properties) is the general number field sieve. The best currently-available implementation of the GNFS consists of a combination of the GGNFS and Msieve projects. It's really the guys behind these tools who deserve the credit for making this possible. While it does take a bit of work to get the tools set up correctly, most of what I did was sitting around waiting for it to finish, and every once in a while, telling the script to try another filtering run.

Some fun statistics:
- The factorization took, in total, about 1745 hours, or a bit less than 73 days, of computation. (I've actually been working on this since early March; I had a couple of false starts and haven't been able to run the software continously.)
- My CPU, for reference, is a dual-core Athlon64 at 1900 MHz.
- The sieving database was 4.9 gigabytes and contained just over 51 million relations.
- During the "filtering" phase, Msieve was using about 2.5 gigabytes of RAM.
- The final processing involved finding the null space of a 5.4 million x 5.4 million matrix.

Oh, and how long have I had this? About two days now. The job finished on Wednesday afternoon, I tested out the result with PongOS, then I came here to tell you all about it.

The other keys will come in their time, I'm sure. If anybody else would like to try factoring one of them, just let me know so we don't step on each other's toes. I haven't started working on any of them yet; I think I'll probably try 0102 next, but I could be persuaded otherwise.

I'm still rather amused that this happened at almost the same time Free83P was released. Great minds think alike...

This post has been edited by **FloppusMaximus**: 31 July 2009 - 07:33 PM

Export-Compatible Key Length

TLS 1.0 «great» cipher suites:

```
TLS_RSA_EXPORT_WITH_RC4_40_MD5         * RSA_EXPORT     RC4_40       MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5     * RSA_EXPORT     RC2_CBC_40   MD5
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA      * RSA_EXPORT     DES40_CBC    SHA
TLS_RSA_WITH_DES_CBC_SHA                 RSA            DES_CBC      SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA   * DH_DSS_EXPORT  DES40_CBC    SHA
TLS_DH_DSS_WITH_DES_CBC_SHA              DH_DSS         DES_CBC      SHA
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA   * DH_RSA_EXPORT  DES40_CBC    SHA
TLS_DH_RSA_WITH_DES_CBC_SHA              DH_RSA         DES_CBC      SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA  * DHE_DSS_EXPORT DES40_CBC    SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA             DHE_DSS        DES_CBC      SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA  * DHE_RSA_EXPORT DES40_CBC    SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA             DHE_RSA        DES_CBC      SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5     * DH_anon_EXPORT RC4_40       MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA    DH_anon        DES40_CBC    SHA
TLS_DH_anon_WITH_DES_CBC_SHA             DH_anon        DES_CBC      SHA
```

DVB-CSA

- Mix of block and stream cipher used to encrypt multimedia streams in the Pay-TV world
- Key size: 48 bits ...
- ... but it changes every 4 to 30 seconds, and one can only hope a ciphertext-only attack in practice.
- Will be replaced by DVB-CSA v3 (128-bit key size)

Right Algorithm

- Every crypto student/engineer should know that textbook-RSA, FEAL-4 and MD4 are to be avoided.
- RSA-{OAEP, PSS}, AES, SHA-256, HMAC, ECDH, ECDSA are good crypto primitives.

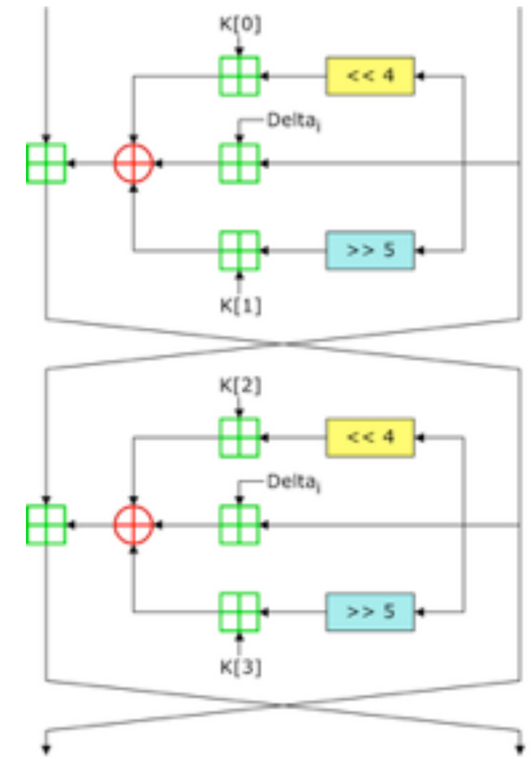
Right Algorithm ?

- TEA and the XBOX
- RC4 and WEP
- MD5, SHA1
- IPSec configurations

TEA and the XBOX hack

TEA and the XBOX hack

- TEA used as a compression function in a home-brew hash function used to perform code authentication at boot time.



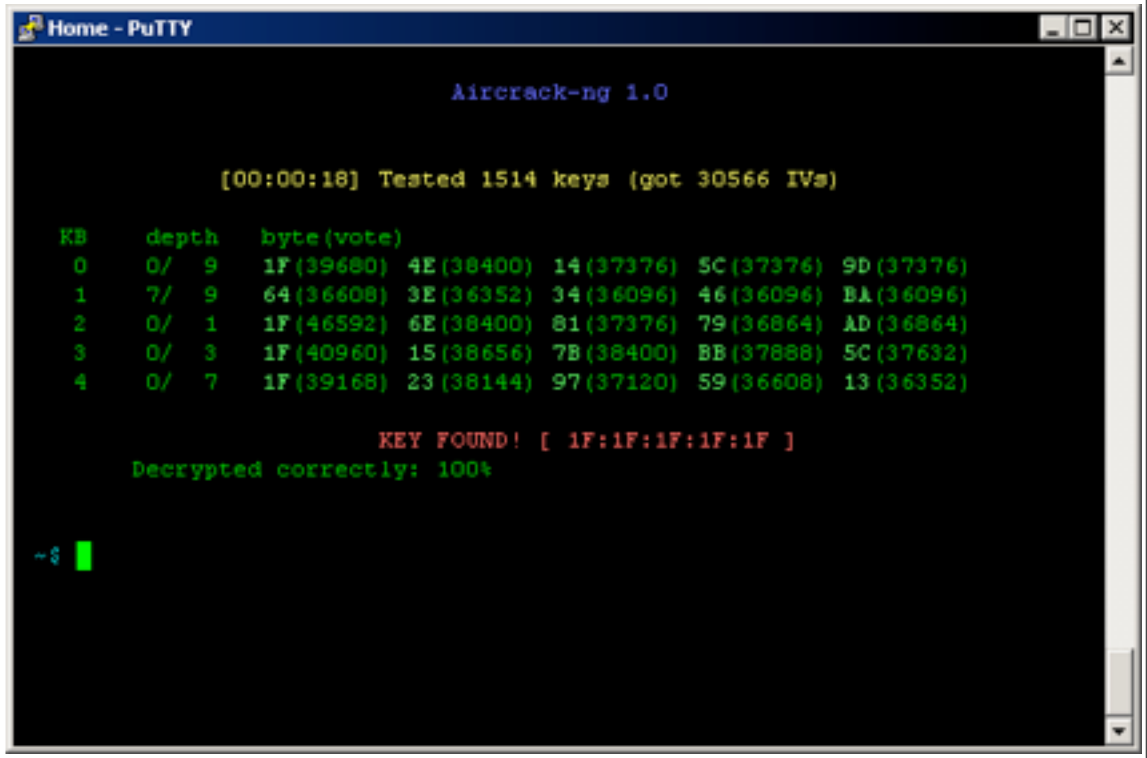
- Unfortunately, in hash mode, equivalent keys = collisions...

Best public cryptanalysis

TEA suffers from equivalent keys (Kelsey et al., 1996) and can be broken using a related-key attack requiring 2^{23} chosen plaintexts and a time complexity of 2^{32}.[1]

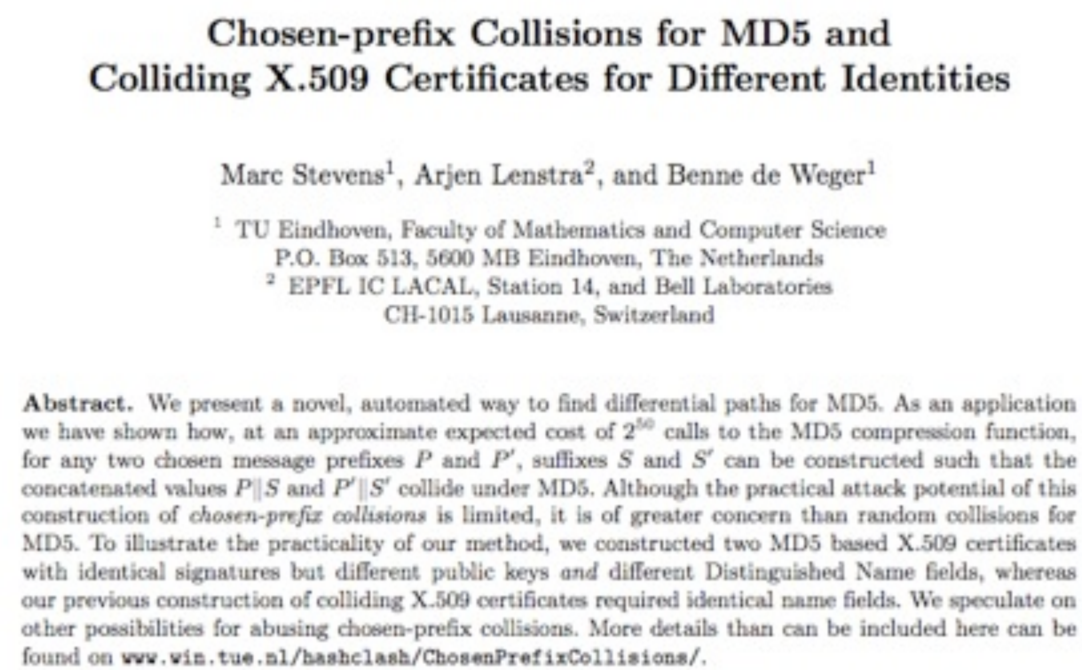
RC4 and WEP

- RC4 used as stream cipher in the wireless network security standard WEP.
- Unfortunately, RC4 suffers from several statistical imperfections at the beginning of its output...



MD5 / SHA1

- MD5 is (still) one of the most widely deployed hash function.
- Unfortunately, it was severely broken in 2004 with respect to its resistance to collisions.
- SHA-1 is ubiquitous in PKI (X.509v3)



Chosen-prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities

Marc Stevens[1], Arjen Lenstra[2], and Benne de Weger[1]

[1] TU Eindhoven, Faculty of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2] EPFL IC LACAL, Station 14, and Bell Laboratories
CH-1015 Lausanne, Switzerland

Abstract. We present a novel, automated way to find differential paths for MD5. As an application we have shown how, at an approximate expected cost of 2^{50} calls to the MD5 compression function, for any two chosen message prefixes P and P', suffixes S and S' can be constructed such that the concatenated values $P\|S$ and $P'\|S'$ collide under MD5. Although the practical attack potential of this construction of *chosen-prefix collisions* is limited, it is of greater concern than random collisions for MD5. To illustrate the practicality of our method, we constructed two MD5 based X.509 certificates with identical signatures but different public keys *and* different Distinguished Name fields, whereas our previous construction of colliding X.509 certificates required identical name fields. We speculate on other possibilities for abusing chosen-prefix collisions. More details than can be included here can be found on `www.win.tue.nl/hashclash/ChosenPrefixCollisions/`.

Right Parameters

- Every crypto student/engineer knows that a random parameter should be random with enough entropy, and not a constant drawn uniformly at random...

Right Parameters ?

- Randomness à la Debian in 2008
  - Broken OpenSSL patch reducing the entropy of its PRNG down to 15 bits
- Sony PS3 and ECDSA
  - Constant instead of a random value in ECDSA used in the secure boot, or how to loose a private key...

Black-Box Adversaries

- This the usual definition of an adversary for cryptographers

Black-Box Adversaries

- Model my algorithm/protocol/system as a set of oracles
- Interact with those oracles
  - Ciphertext-only
  - Known plaintext-ciphertext
  - Chosen (adaptively or not) plaintexts and/or ciphertexts

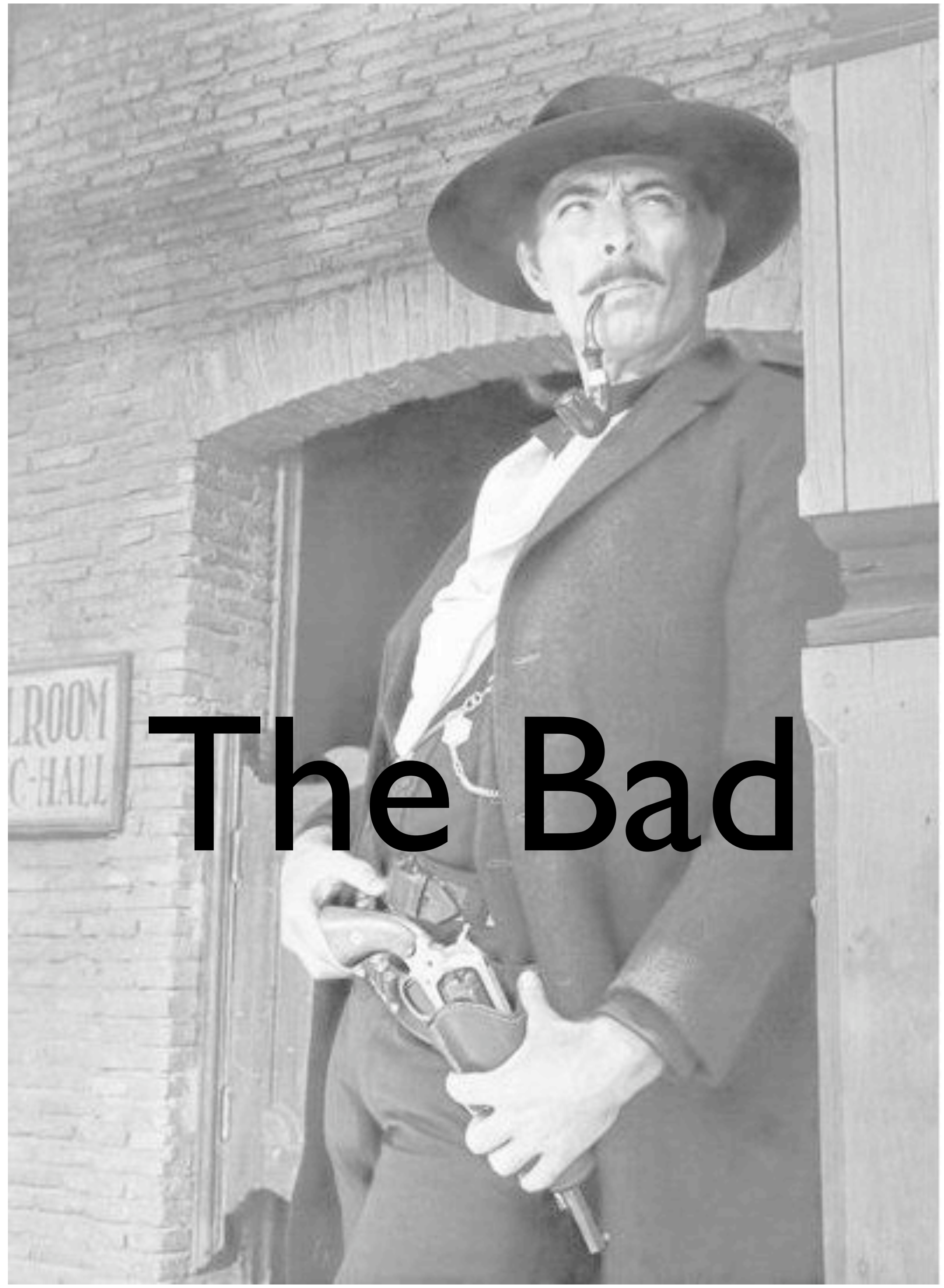
Black-Box Adversaries

- Prove (mathematically) that your algorithm/protocol/system is secure if the underlying cryptographic primitives are secure.
- Examples:
  - RSA-OAEP
  - RSA-PSS

In Summary...

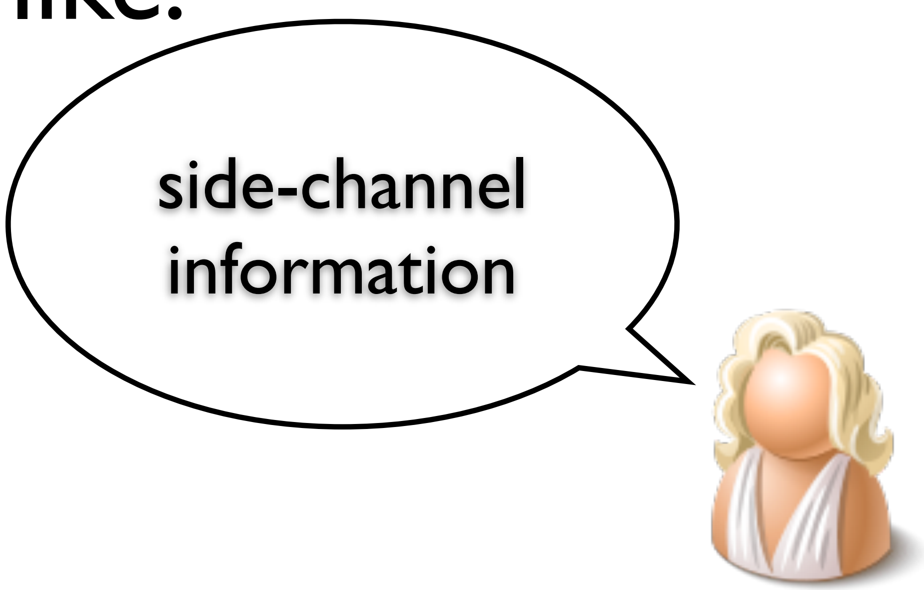
- With respect to black-box adversaries, people tend to do it right, although it is possible to find many, many counter-exemples...
- Cryptographers have done a good job teaching and explaining dangers.

The Bad



Grey-Box Adversaries

- Adversaries that were NOT foreseen by cryptographers...
- Can interact with the cryptographic primitives, but might have (just) a bit more information about the computations, like:
  - Timings
  - Physical leakage
  - Faults



Side-Channel Attacks

- Timing
- Physical Leakage
- Faults

Timing Attacks

Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher

Cryptography Research, Inc.
607 Market Street, 5th Floor, San Francisco, CA 94105, USA.
E-mail: paul@cryptography.com.

Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext. Actual systems are potentially at risk, including cryptographic tokens, network-based cryptosystems, and other applications where attackers can make reasonably accurate timing measurements. Techniques for preventing the attack for RSA and Diffie-Hellman are presented. Some cryptosystems will need to be revised to protect against the attack, and new protocols and algorithms may need to incorporate measures to prevent timing attacks.

Keywords: timing attack, cryptanalysis, RSA, Diffie-Hellman, DSS.

Timing Attacks

FIGURE 1: RSAREF Modular Multiplication Times

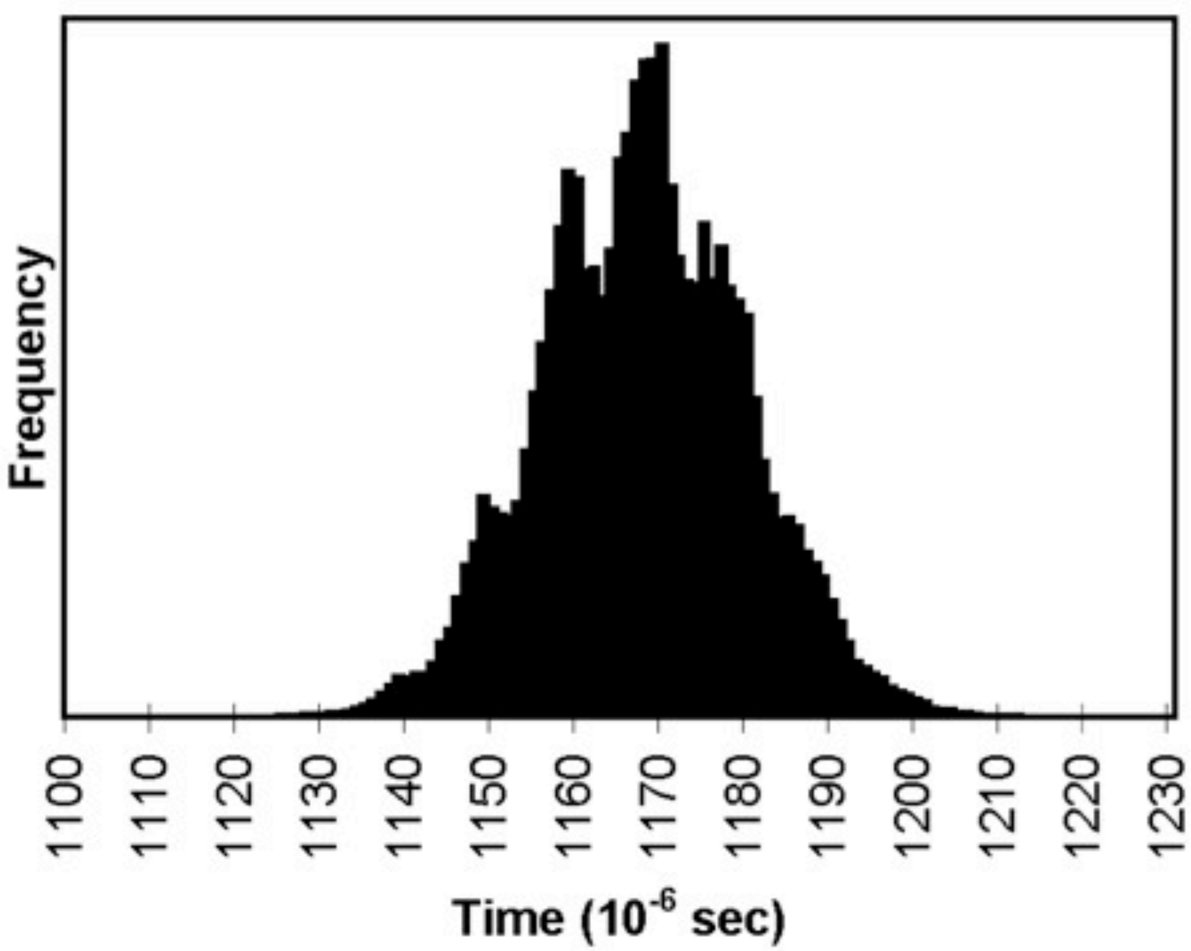
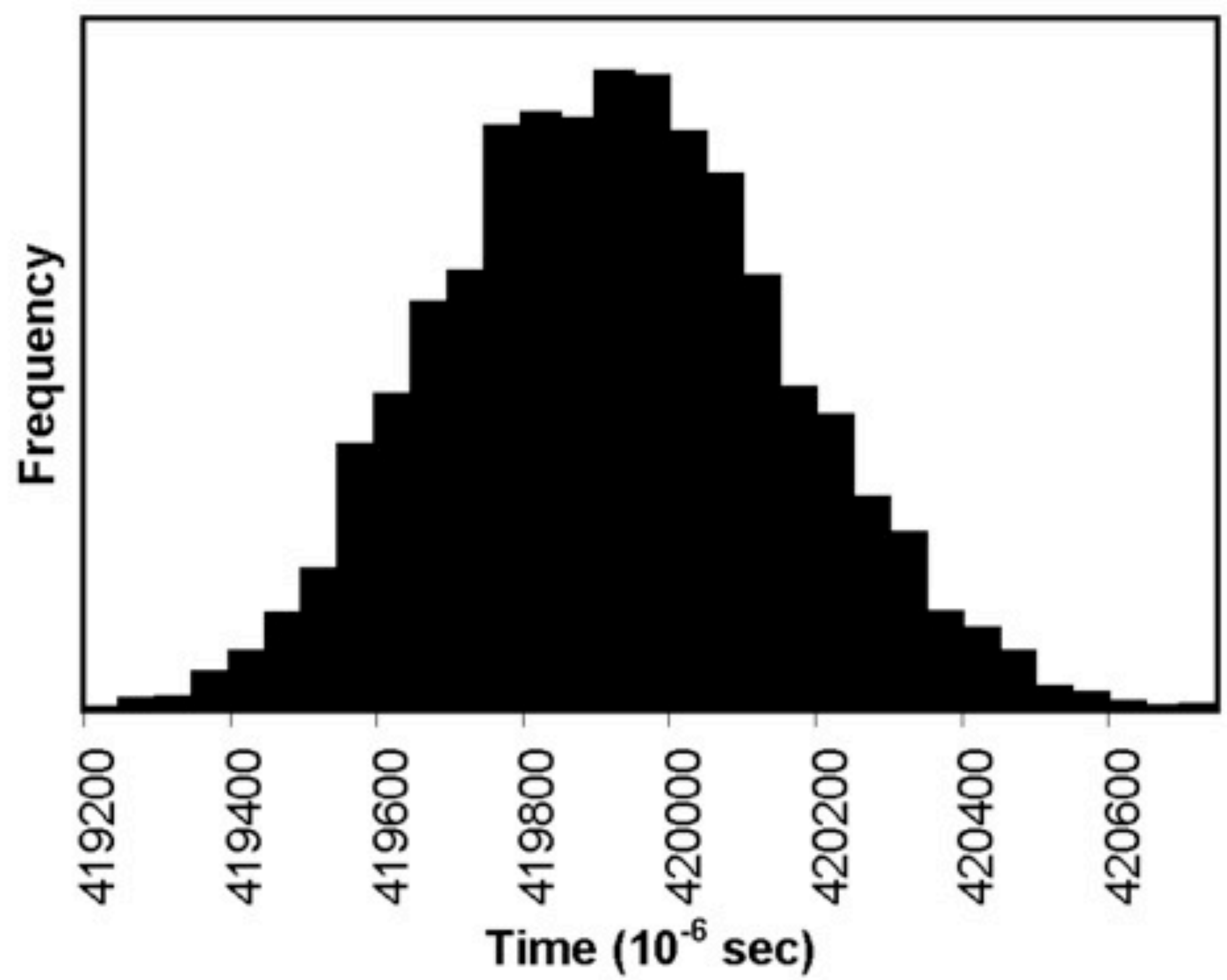
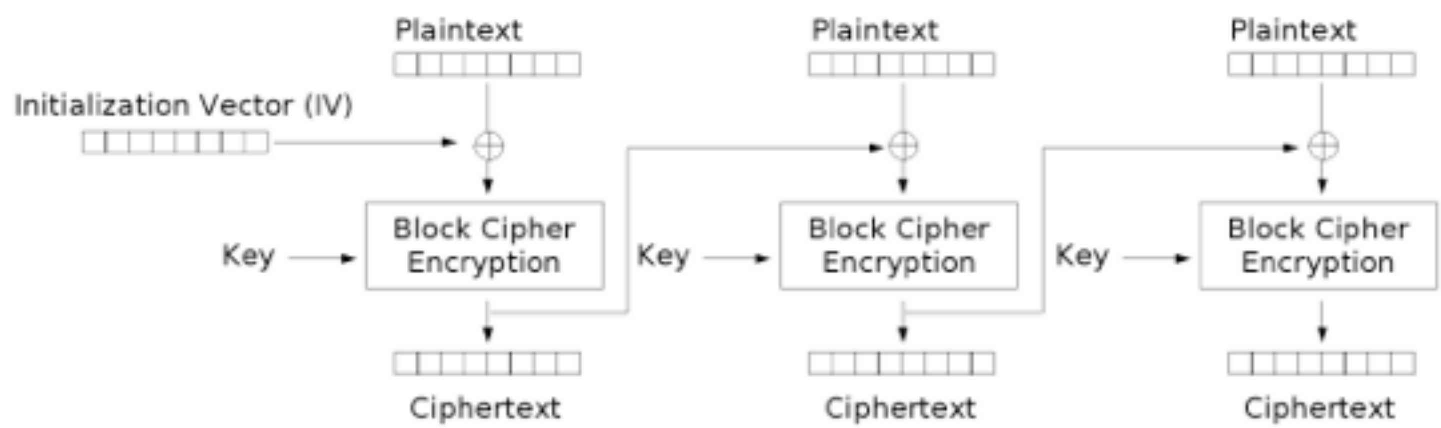


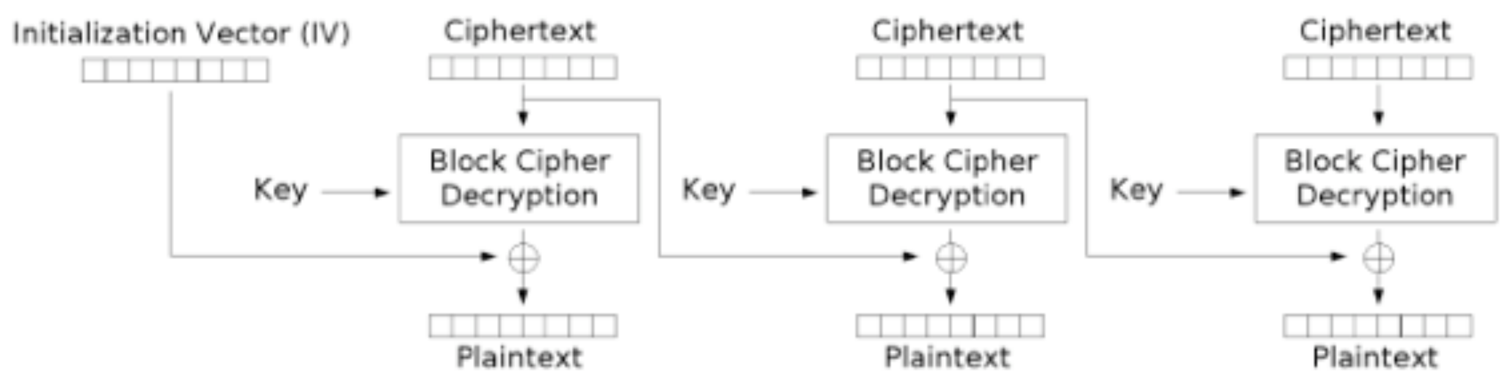
FIGURE 2: RSAREF Modular Exponentiation Times



Timing Attacks



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Timing Attacks

- Standard padding with 8-bytes blocks:
  - Missing 3 bytes: pad with `03 03 03`
  - Missing 7 bytes: pad with `07 07 07 07 07 07 07`
  - Missing 0 bytes: pad with `08 08 08 08 08 08 08 08`

Timing Attacks

- Problem (spotted in 2002, exploited in 2003) if the padding checking routine is not time-constant:

Password Interception in a SSL/TLS Channel

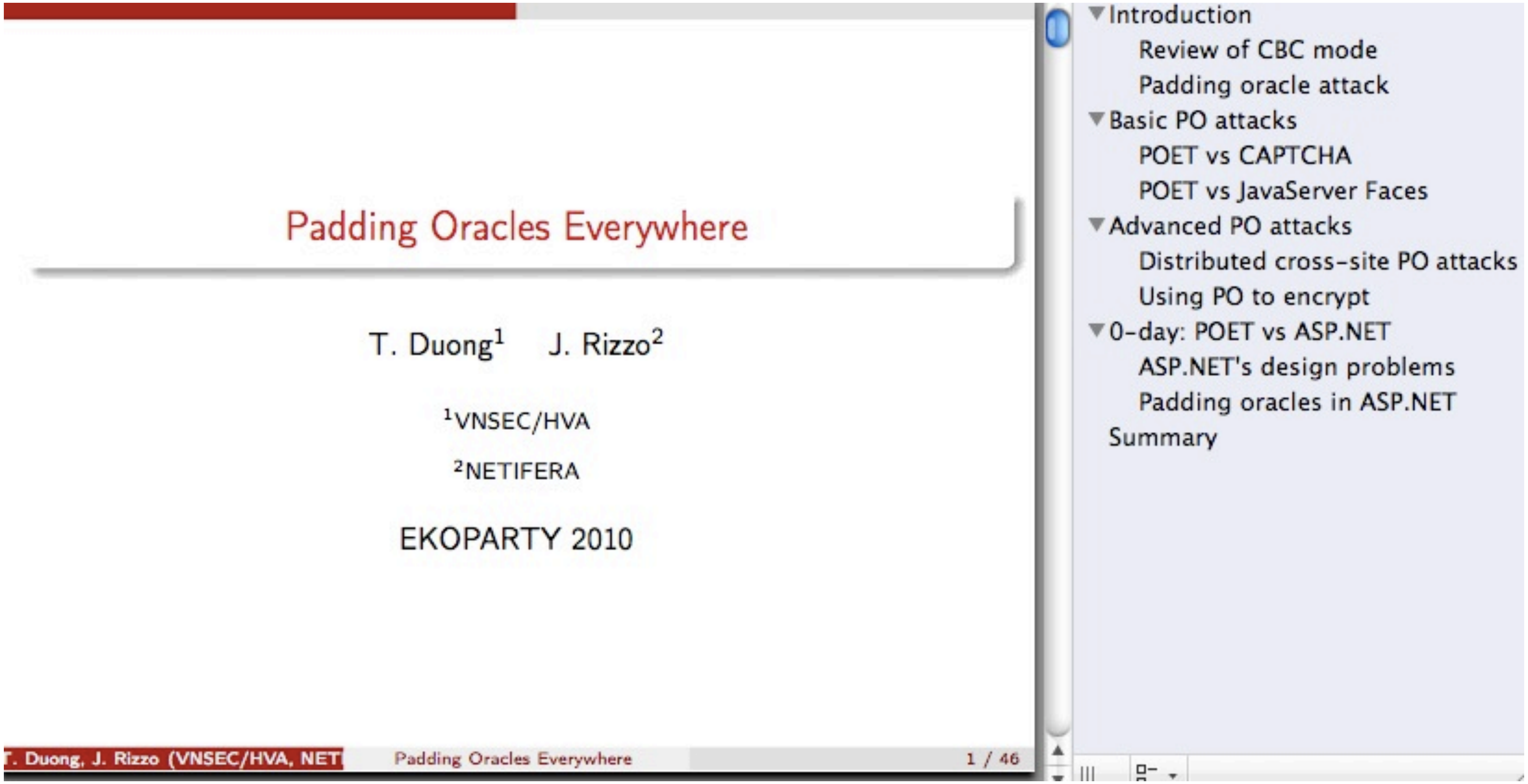
Brice Canvel[1], Alain Hiltgen[2], Serge Vaudenay[1], and Martin Vuagnoux[3]

[1] Swiss Federal Institute of Technology (EPFL) - LASEC
http://lasecwww.epfl.ch
[2] UBS AG
email:alain.hiltgen@ubs.com
[3] EPFL - SSC, and Ilion
http://www.ilionsecurity.ch

Abstract. Simple password authentication is often used e.g. from an email software application to a remote IMAP server. This is frequently done in a protected peer-to-peer tunnel, e.g. by SSL/TLS.
At Eurocrypt'02, Vaudenay presented vulnerabilities in padding schemes used for block ciphers in CBC mode. He used a side channel, namely error information in the padding verification. This attack was not possible against SSL/TLS due to both unavailability of the side channel (errors are encrypted) and premature abortion of the session in case of errors. In this paper we extend the attack and optimize it. We show it is actually applicable against latest and most popular implementations of SSL/TLS (at the time this paper was written) for password interception.
We demonstrate that a password for an IMAP account can be intercepted when the attacker is not too far from the server in less than an hour in a typical setting.
We conclude that these versions of the SSL/TLS implementations are not secure when used with block ciphers in CBC mode and propose ways to strengthen them. We also propose to update the standard protocol.

Timing Attacks

- Padding oracles reloaded (but here, not based on timing):



Timing Attacks

- Cache attacks:

Cache Attacks and Countermeasures: the Case of AES

(Extended Version)

revised 2005-11-20

Dag Arne Osvik[1], Adi Shamir[2] and Eran Tromer[2]

[1] dag.arne@osvik.no

[2] Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel
{adi.shamir, eran.tromer}@weizmann.ac.il

Abstract. We describe several software side-channel attacks based on inter-process leakage through the state of the CPU's memory cache. This leakage reveals memory access patterns, which can be used for cryptanalysis of cryptographic primitives that employ data-dependent table lookups. The attacks allow an unprivileged process to attack other processes running in parallel on the same processor, despite partitioning methods such as memory protection, sandboxing and virtualization. Some of our methods require only the ability to trigger services that perform encryption or MAC using the unknown key, such as encrypted disk partitions or secure network links. Moreover, we demonstrate an extremely strong type of attack, which requires knowledge of neither the specific plaintexts nor ciphertexts, and works by merely monitoring the effect of the cryptographic process on the cache. We discuss in detail several such attacks on AES, and experimentally demonstrate their applicability to real systems, such as OpenSSL and Linux's `dm-crypt` encrypted partitions (in the latter case, the full key can be recovered after just 800 writes to the partition, taking 65 milliseconds). Finally, we describe several countermeasures which can be used to mitigate such attacks.

Timing Attacks

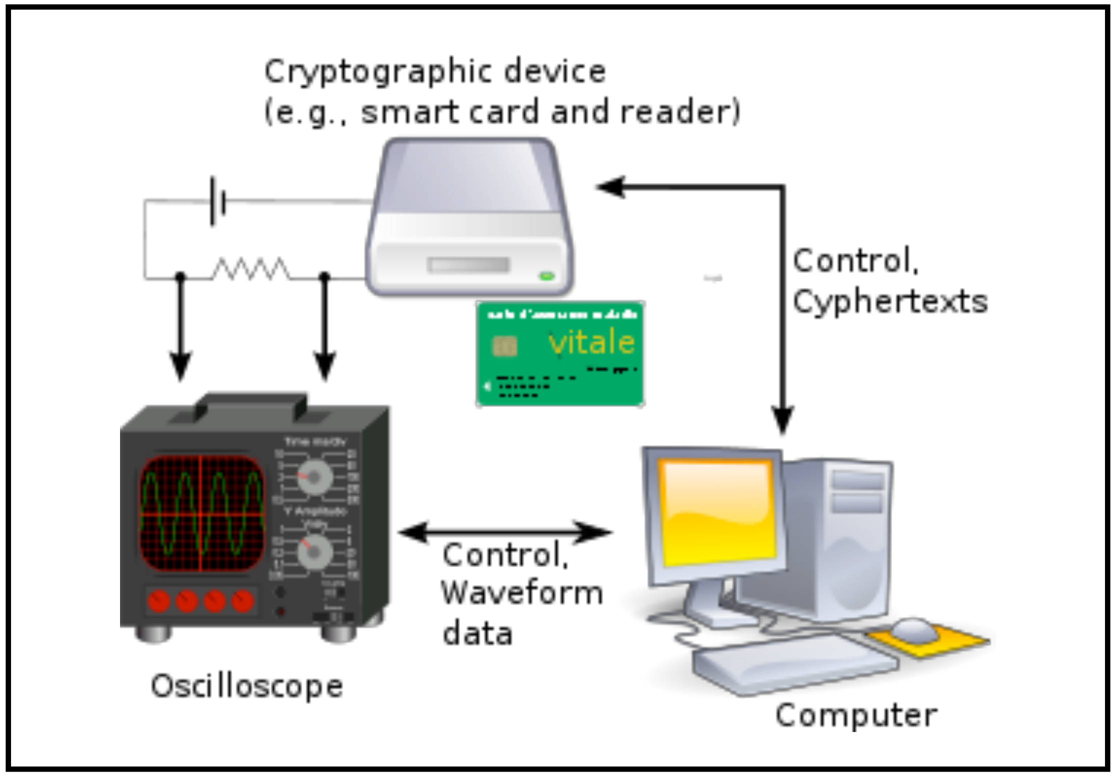
- IDEA's multiplication in GF(65537)
  - `0x0000 == 0x10000`
- Schneier et al. timing attack in $O(2^{32})$ ops
- OpenSSL code:

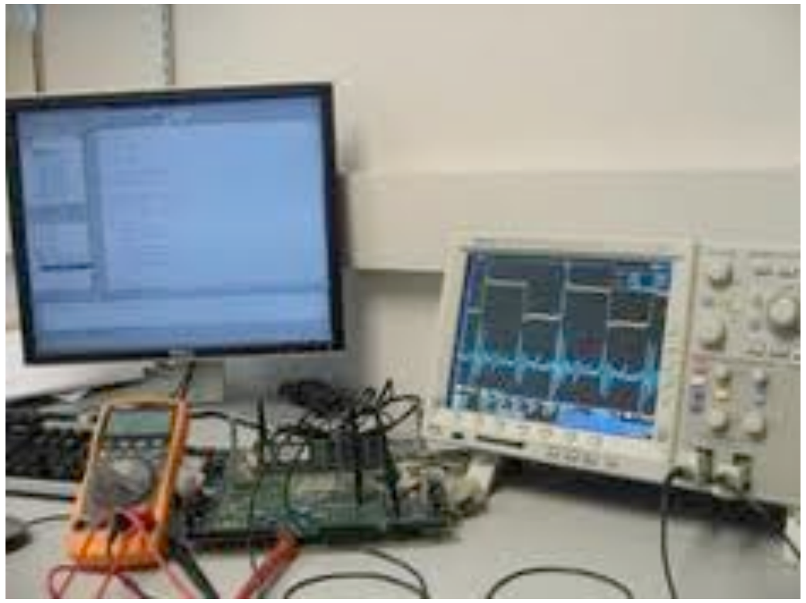
```
#define idea_mul(r,a,b,ul) \
ul=(unsigned long)a*b; \
if (ul != 0) \
        { \
        r=(ul&0xffff)-(ul>>16); \
        r-=((r)>>16); \
        } \
else \
        r=(-(int)a-b+1); /* assuming a or b is 0 and in range */
```

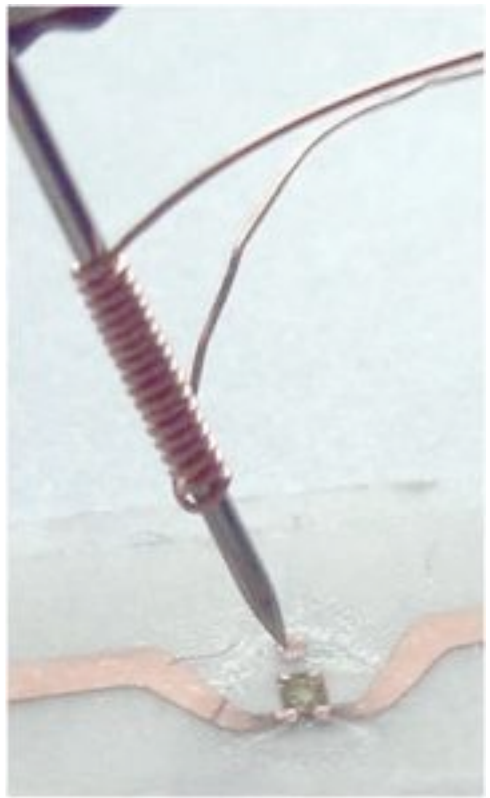
Attacks based on Physical Leakage

- As a matter of fact, computations executed on any kind of platform (SW/HW) consumes energy...
- If it is possible to measure this energy, and if this energy consumption is dependent on secret values, then those secret are at risk !

Attacks based on Physical Leakage







Attacks based on Physical Leakage

Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun

Cryptography Research, Inc.
607 Market Street, 5th Floor
San Francisco, CA 94105, USA.
http://www.cryptography.com
E-mail: {paul,josh,ben}@cryptography.com.

Abstract. Cryptosystem designers frequently assume that secrets will be manipulated in closed, reliable computing environments. Unfortunately, actual computers and microchips leak information about the operations they process. This paper examines specific methods for analyzing power consumption measurements to find secret keys from tamper resistant devices. We also discuss approaches for building cryptosystems that can operate securely in existing hardware that leaks information.

Keywords: differential power analysis, DPA, SPA, cryptanalysis, DES

Attacks based on Physical Leakage

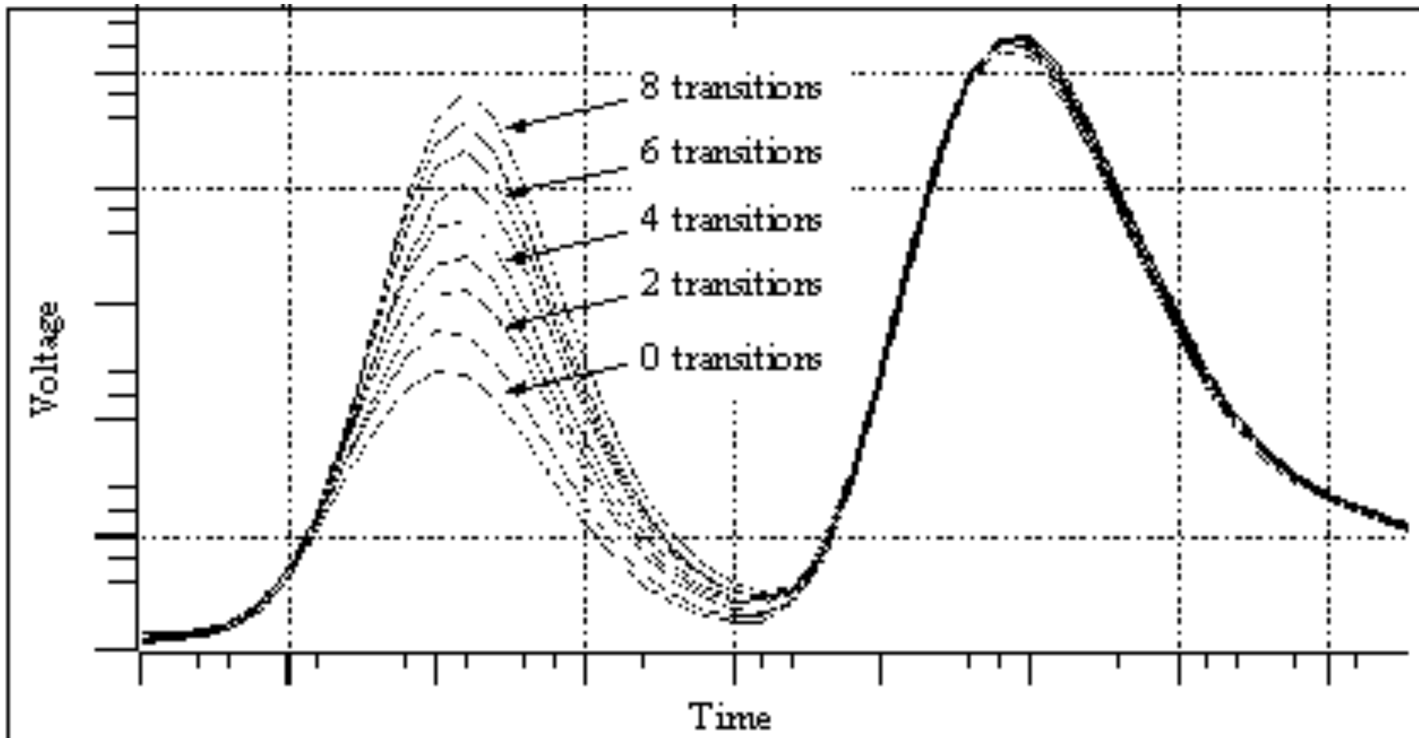


FIGURE 2. **Number of Bit Transitions versus Power Consumption**
These results show how the data effects the power levels. The nine overlayed waveforms correspond to the power traces of different data being accessed by an LDA instruction. These results were obtained by averaging the power signals across 500 samples in order to reduce the noise content. The difference in voltage between i transitions and i+1 transitions is about 6.5 mV.

Attacks based on Faults

- Consider the following piece of code that could valid the RSA signature during the secure boot of a trusted device:

```
if (RSA_verify (signature) ==
RSA_VALID_SIGNATURE) {
  // Perform some critical operation
  } else {
   return NOT_AUTHENTICATED
  }
```

Attacks based on Faults

- This could translate into the following:

The whole RSA signature verification mechanism security relies on whether this instruction will be executed or not...

```
...
cmp    $0x0, %ebx
jne    0x64FE89A1
...
```

Attacks based on Faults

Design Principles for Tamper-Resistant Smartcard Processors

Oliver Kömmerling
Advanced Digital Security Research
Mühlstraße 7
66484 Riedelberg
Germany
ok@adsr.de

Markus G. Kuhn
University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge CB2 3QG
United Kingdom
mgk25@cl.cam.ac.uk

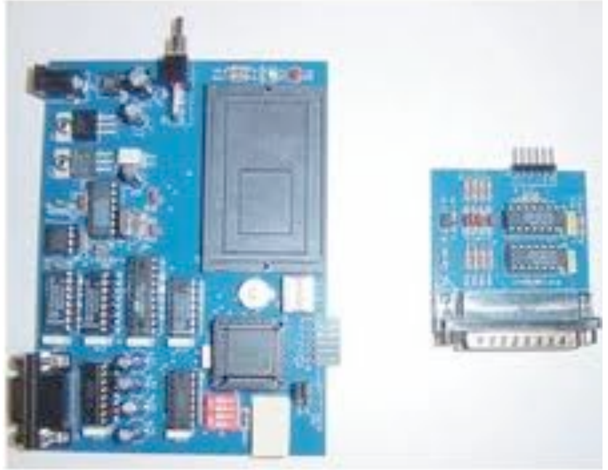
2.2.1 Glitch Attacks

In a glitch attack, we deliberately generate a malfunction that causes one or more flipflops to adopt the wrong state. The aim is usually to replace a single critical machine instruction with an almost arbitrary other one. Glitches can also aim to corrupt data values as they are transferred between registers and memory. Of the many fault-induction attack techniques on smartcards that have been discussed in the recent literature [11, 12, 16, 17, 18], it has been our experience that glitch attacks are the ones most useful in practical attacks.

We are currently aware of three techniques for creating fairly reliable malfunctions that affect only a very small number of machine cycles in smartcard processors: clock signal transients, power supply transients, and external electrical field transients.

Particularly interesting instructions that an attacker might want to replace with glitches are conditional jumps or the test instructions preceding them. They create a window of vulnerability in the processing stages of many security applications that often allows us to bypass sophisticated cryptographic barriers by simply preventing the execution of the code that detects that an authentication attempt was unsuccessful. Instruction glitches can also be used to extend the runtime of loops, for instance in serial port output routines to see more of the memory after the output buffer [12], or also to reduce the runtime of loops, for instance to transform an iterated cipher function into an easy to break single-round variant [11].

Attacks based on Faults



Correct Implementation

- OpenSSL and bug attacks

Practical realisation and elimination of an ECC-related software bug attack*

B. B. Brumley[1], M. Barbosa[2], D. Page[3], and F. Vercauteren[4]

[1] Department of Information and Computer Science,
Aalto University School of Science, P.O. Box 15400, FI-00076 Aalto, Finland.
billy.brumley@aalto.fi
[2] HASLab/INESC TEC
Universidade do Minho, Braga, Portugal.
mbb@di.uminho.pt
[3] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK.
page@cs.bris.ac.uk
[4] Department of Electrical Engineering, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.
fvercaut@esat.kuleuven.ac.be

Abstract. We analyse and exploit implementation features in OpenSSL version 0.9.8*g* which permit an attack against ECDH-based functionality. The attack, although more general, can recover the entire (static) private key from an associated SSL server via 633 adaptive queries when the NIST curve P-256 is used. One can view it as a software-oriented analogue of the bug attack concept due to Biham et al. and, consequently, as the first bug attack to be successfully applied against a real-world system. In addition to the attack and a posteriori countermeasures, we show that formal verification, while rarely used at present, is a viable means of detecting the features which the attack hinges on. Based on the security implications of the attack and the extra justification posed by the possibility of intentionally incorrect implementations in collaborative software development, we conclude that applying and extending the coverage of formal verification to augment existing test strategies for OpenSSL-like software should be deemed a worthwhile, long-term challenge.

Keywords: elliptic curve, OpenSSL, NIST, fault attack, bug attack.

OpenSSL and Sisters

- Several general-purpose open-source cryptographic libraries do exist (non-exhaustive list):
  - OpenSSL
  - libgcrypt
  - Mozilla NSS
  - libtomcrypt
  - NaCl
  - Botan
  - Crypto++
  - cryptlib

OpenSSL and Sisters

- Natural question asked less than one year ago :
  - How secure are general-purpose open-source cryptographic libraries ?

OpenSSL and Sisters

- What means «security» here ?
  - Resistance to well-known cryptographic attacks
  - Resistance to side-channel attacks
  - (Respect of best practices in terms of secure programming)
  - (Reactivity of its developers when confronted to security issues)
  - ...

Manger's Attack

- Published by James Manger at Crypto'01
- Attack bad implementations of RSA-OAEP padding mechanisms
- Transform a «bad» implementation into a decryption oracle.
- Requires only about 1024 adaptively chosen queries to decrypt a 1024-bit RSA ciphertext

Manger’s Attack

- Only required information: «Does the decrypted ciphertext has the most significant byte equal to `0x00` ?»

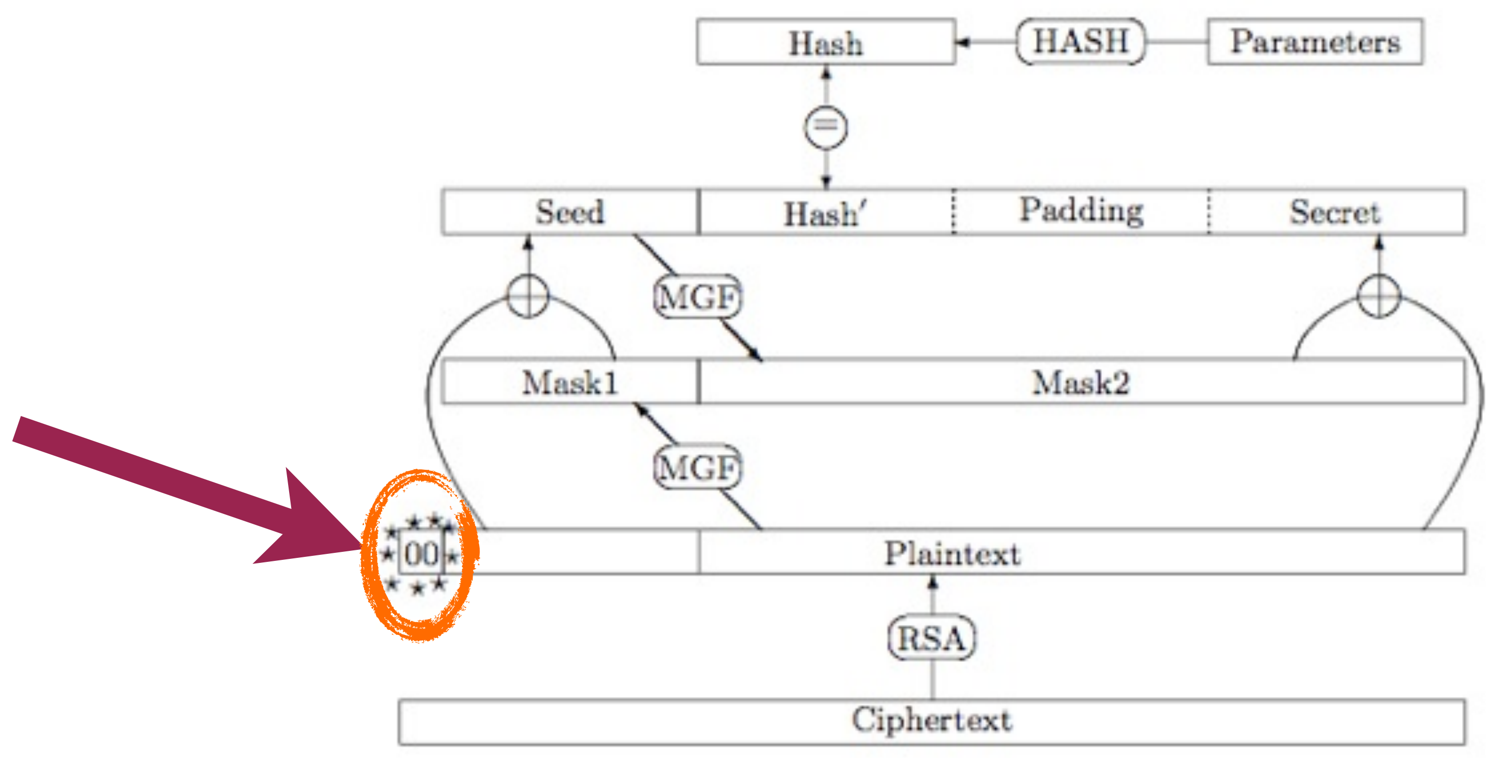


Fig. 1. RSAES-OAEP Decoding

Manger's Attack

- One can obtain this information (at least) through
  - Error messages
  - Timing differences

Manger's Attack

- Let's have a look at OpenSSL's implementation:

```
CHANGES

*) Improve RSA_padding_check_PKCS1_OAEP() check
again to avoid 'wristwatch attack' using huge
encoding parameters (cf. James H. Manger's CRYPTO
2001 paper). Note that the RSA_PKCS1_OAEP_PADDING
case of RSA_private_decrypt() does not use encoding
parameters and hence was not vulnerable.[Bodo
Moeller]
```

Manger's Attack

- Further:

```
/* crypto/rsa/rsa_oaep.c */
...
/* signalling this error immediately after detection
 * might allow for side-channel attacks (e.g. timing
 * if 'plen' is huge -- cf. James H. Manger, "A
 * Chosen Ciphertext Attack on RSA Optimal
 * Asymmetric Encryption Padding (OAEP) [...]",
 * CRYPTO 2001), so we use a 'bad' flag */
```

Manger's Attack

- However...

```
if (lzero < 0)
        {
/* signalling this error immediately after
 * detection might allow
 * for side-channel attacks (e.g. timing if
 * 'plen' is huge
 * -- cf. James H. Manger, "A Chosen
 * Ciphertext Attack on RSA Optimal
 * Asymmetric Encryption Padding (OAEP)
 * [...]", CRYPTO 2001),
 * so we use a 'bad' flag */
bad = 1;
lzero = 0;
flen = num; /* don't overflow the memcpy to
             *padded_from */
}
```

Manger's Attack

- Out of NaCl's homepage:

The CPU's instruction pointer, branch predictor, etc. are not designed to keep information secret. For performance reasons this situation is unlikely to change. The literature has many examples of successful timing attacks that extracted secret keys from these parts of the CPU.

Manger's Attack

- Is that time-constant ?
- Time to compute 1'048'576 checks on my MacBook Pro:

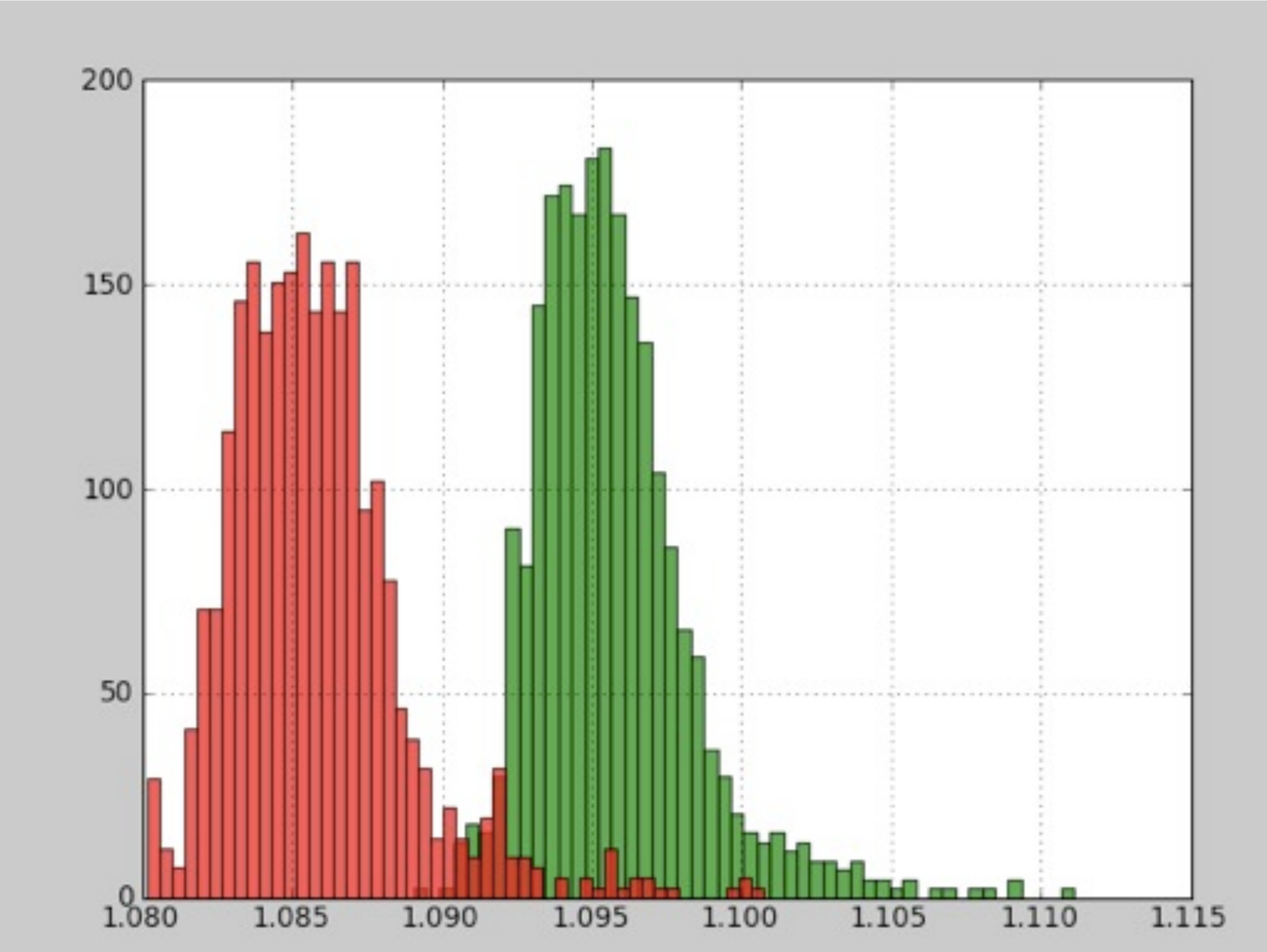
```
macbook-pro-de-pascal-junod:openssl_manger pjunod$ ./junk

[VALID PADDING (20971520)   ] : 10.943075 seconds for
1048576 OAEP check

[INVALID PADDING (-1048576) ] : 10.835983 seconds for
1048576 OAEP checks
```

Manger's Attack

- Distribution of 1000 independent measures of 104'858 checks



Manger's Attack

- Is OpenSSL broken (with respect to Manger's attack) ?
  - On high-end servers/desktop
    - In theory, yes !
    - In practice, the number of measurement required to remove the noise (due to networking mainly) is probably too large...

Manger’s Attack

- Is OpenSSL broken (with respect to Manger’s attack) ?
  - On embedded platforms:
    - YES, DEFINITIVELY !!
    - Clock-cycle accurate measurement is possible.
    - If time-constant, use the power trace of the execution.

Botan

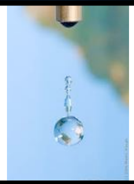
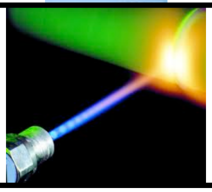
```
// Is this vulnerable to timing attacks?
   for(u32bit i = HASH_LENGTH + Phash.size(); i !=
tmp.size(); ++i)
      {
      if(tmp[i] && !delim_idx)
         {
         if(tmp[i] == 0x01)
            delim_idx = i;
         else
            delim_ok = false;
         }
      }
```

Crypto++

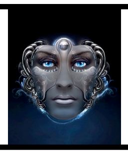
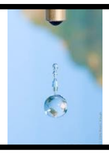
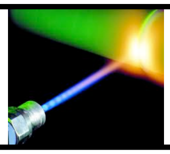
```
bool invalid = false;

// convert from bit length to byte length
if (oaepBlockLen % 8 != 0)
    {
        invalid = (oaepBlock[0] != 0) ||invalid;
        oaepBlock++;
    }
```

Legend

	Classical timing attacks
	Cache attacks
	Oracle attacks
	Leakage attacks
	Fault attacks
✔	Serious care
~	Some care, but not always/properly
✖	No care at all

Summary

					
OpenSSL	✔	~	~	✖	✖
libgcrypt	✔	✖	✖	✖	✖
libtomcrypt	✖	✖	✖	✖	✖
NSS	✔	✖	~	✖	~
NaCl	✔	✔	✔	✖	✖
Botan	✔	✔	~	✖	~
Crypto++	✔	~	~	✖	✖
cryptlib	✔	✖	~	~	~

In Summary...

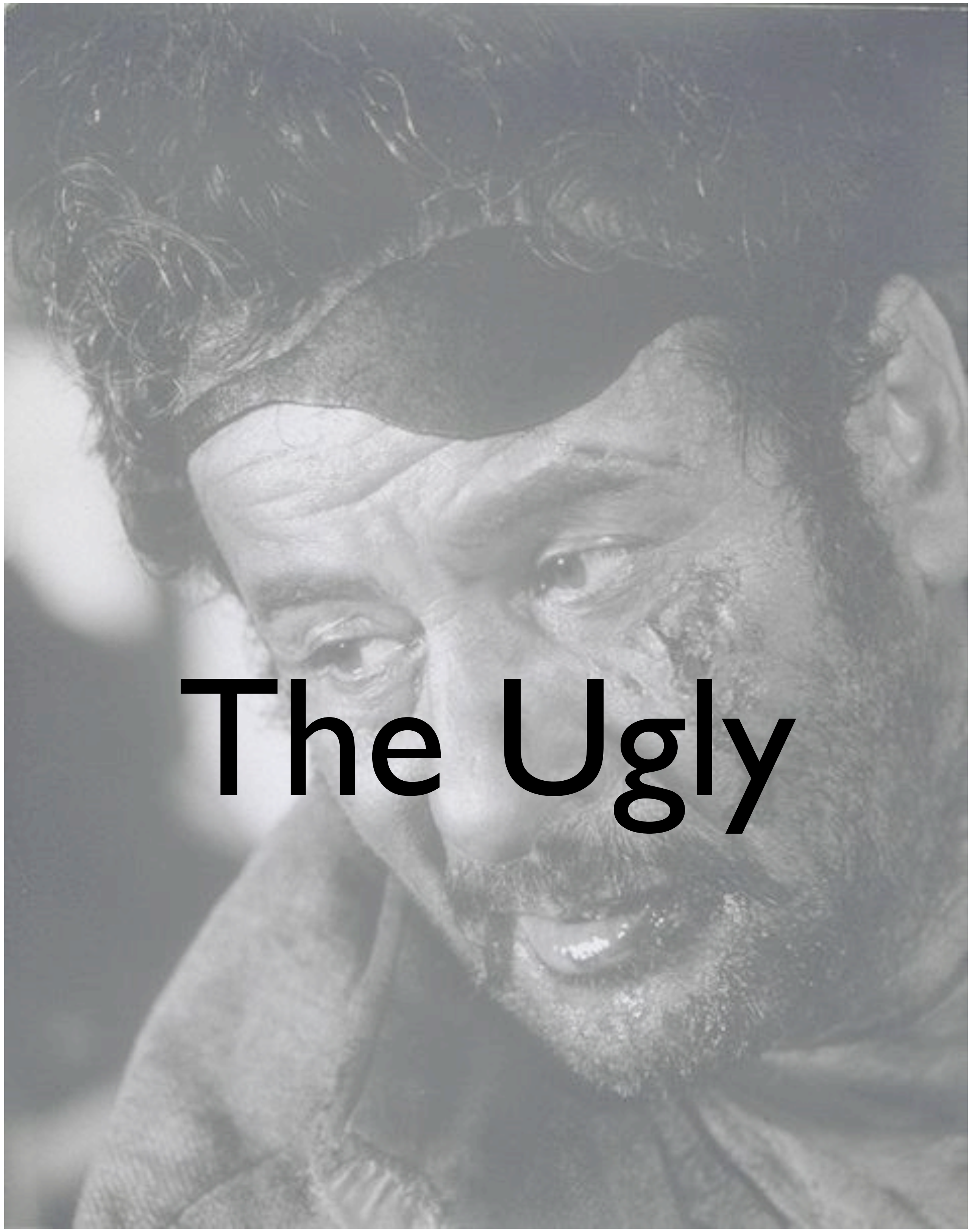
- Most of open-source, free crypto libraries are not protected against side-channel atacks.
- A prominent counter-example is NaCl, which is time-constant and probably oracle-free... but it does not implement standard crypto !

In Summary...

- While the silicon industry is aware of the side-channel problem (probably because they all had to pay for implementing protections ;-), the open-source world is **not**.
- Keyword: protection schemes, leakage resilient cryptography (?!)

In Summary...

- Lightweight crypto allows to reinvest the gained Boolean gates to implement side-channel protection schemes !

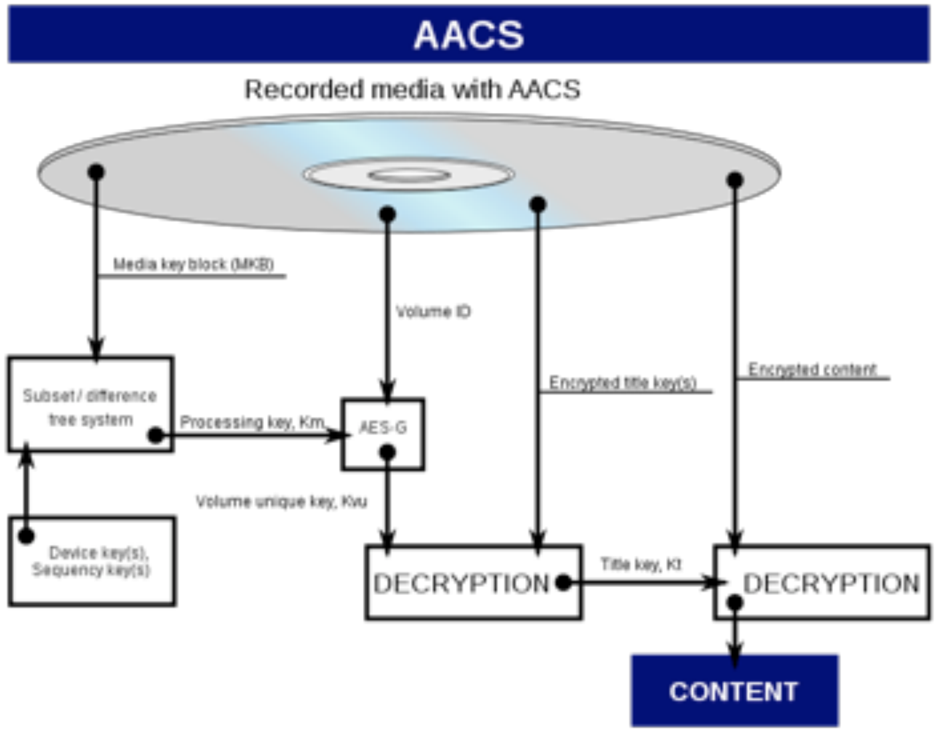

The Ugly

White-Box Adversaries

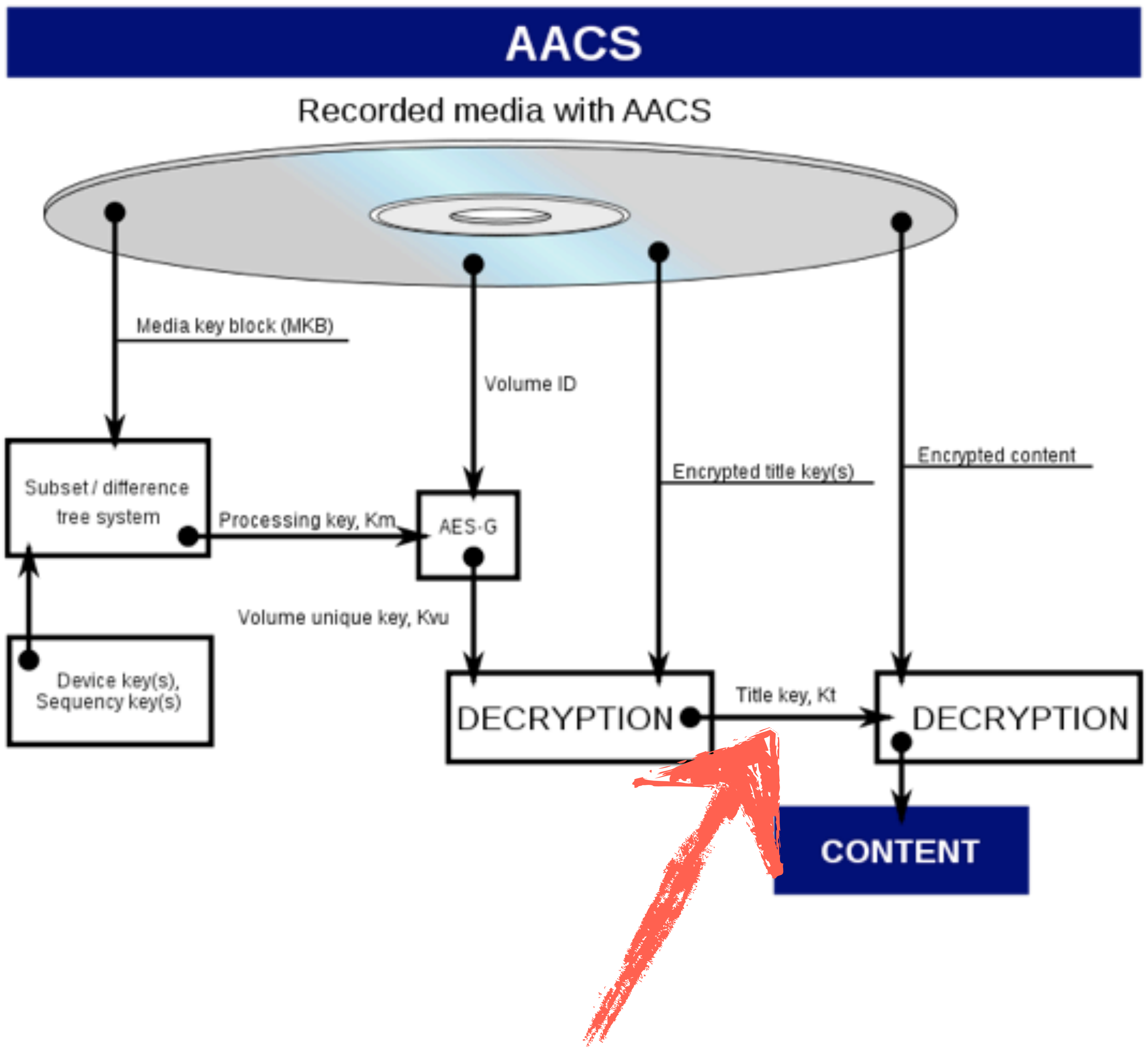
- Adversaries that most practical cryptographers just do not want to hear about...
- Can do EVERYTHING they want !!
  - Complete reverse-engineering of SW/HW
  - Read/Write all memories, including secure ones (containing keys)
  - Perturb all computations

White-Box Adversaries

- One example among many others: the AACS hack
  - DRM protection scheme for Blu-Ray
  - State-of-the art crypto
  - SW player broken
    - Just read the last media-encrypting key in memory



White-Box Adversaries



Broadcast Encryption and Traitor Tracing

- Broadcast encryption (BE) is a cryptographic technique that allows to selectively define who is able to decrypt a given global ciphertext.
- Traitor tracing (TT) allows to identify pirate sharing their decryption material.

- BE + TT are used to encrypt global symetrical sessions keys, which do not resist white-box adversaries... (cf. CW sharing)

In Summary...

- Resisting white-box adversaries is still a wide OPEN problem.
- White-box cryptography (see Wyseur's PhD thesis)
- Problem tackled by theory-loving authors, but I'm not sure about the current practical significance of their work.
- Keyword: obfuscation

Thank You !

- Website `http://crypto.junod.info`
- Twitter `@cryptopathe`
- E-mail `pascal@junod.info`