Open-Source Cryptographic Libraries and Embedded Platforms

Pascal Junod // HEIG-VD





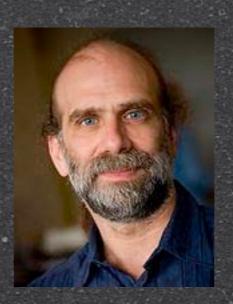
Outline

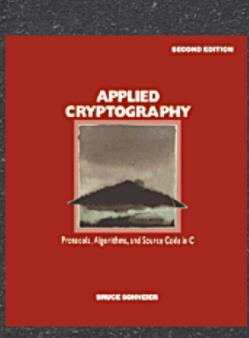
- Cryptography and Security
- Side-Channel Attacks
- What about OpenSSL and Sisters ?

Cryptography and Security

Cryptography Everywhere

With the apparition of Internet and other networks, the use of cryptography has exploded.





This new edition of the cryptography classic provides you with a comprehensive survey of modern cryptography. The book details how programmers and electronic communications professionals can use cryptography -- the technique of enciphering and deciphering messages -- to maintain the privacy of computer data. It describes dozens of cryptography algorithms, gives practical advice on how to implement them in cryptographic software, and shows how they can be used to solve security problems. Covering the latest developments in practical cryptographic techniques, this new edition shows programmers who design computer applications, networks, and storage systems how they can build security into their software and systems.

But...

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 sending information across a network, seen 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 nonest mistake; for this issue, though, customers...MORE >>

[19] CWE-306: Missing Authentication for Critical Function

In countless action movies, the vinam breaks into a high-security balloning 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

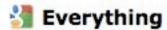
You may be tempted to develop your own encryption scheme in the hopes or making it difficult for attackers to crack. This kind of grow-your-own cryptography is a welcome sight to attackers...MORE >>

What might go wrong?

- Bad choice of primitive
 - TEA and the XBOX hack
 - RC4 and WEP
 - MD5

TEA and the XBOX hack









▼ More

All results

Wonder wheel Sites with images

More search tools

tiny encryption algorithm

tiny encryption algorithm

tiny encryption algorithm java

tiny encryption algorithm python

tiny encryption algorithm code

tiny encryption

About 34,500 results (0.19 seconds)

Tiny Encryption Algorithm - Wikipedia, the free encyclopedia

In cryptography, the Tiny Encryption Algorithm (TEA) is a block cipher notable for its simplicity of description and implementation, typically a few lines ...

Properties - Versions - Reference code - See also

en.wikipedia.org/wiki/Tiny_Encryption_Algorithm - Cached - Similar

[PDF] TEA, a Tiny Encryption Algorithm 9

File Format: PDF/Adobe Acrobat - Quick View



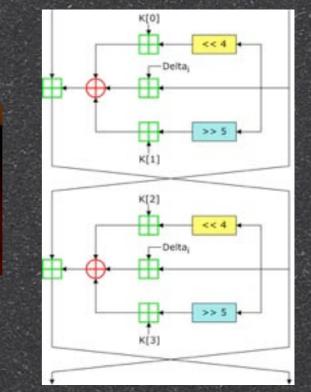
Search

I'm Feeling Lucky »

Advanced search

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 2²³ chosen plaintexts and a time complexity of 2³².^[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...

```
### Home - PuTTY

Aircrack-ng 1.0

[00:00:18] Tested 1514 keys (got 30566 IVs)

KB depth byte (vote)
0 0/ 9 1F (39680) 4E (38400) 14 (37376) 5C (37376) 9D (37376)
1 7/ 9 64 (36608) 3E (36352) 34 (36096) 46 (36096) BA (36096)
2 0/ 1 1F (46592) 6E (30400) 81 (37376) 79 (36664) AD (36664)
3 0/ 3 1F (40960) 15 (38656) 78 (38400) 8B (37888) 5C (37632)
4 0/ 7 1F (39168) 23 (38144) 97 (37120) 59 (36608) 13 (36352)

KEY FOUND! [ 1F:1F:1F:1F:1F ]

Decrypted correctly: 100%
```

MD5

- 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.

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

Marc Stevens¹, Arjen Lenstra², and Benne de Weger¹

 TU Eindhoven, Faculty of Mathematics and Computer Science P.O. Box 513, 5600 MB Eindhoven, The Netherlands
 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 2⁵⁰ 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/.

What might go wrong?

- Bad choice of protocol
 - IPsec in encrypt-only mode

IPsec in Encrypt-Only Mode

- IPsec allows to setup several (too much...) different security configurations:
 - Encrypt only
 - Authenticate only
 - Encrypt and Authenticate

IPsec in Encrypt-Only Mode

Attacking the IPsec Standards in Encryption-only Configurations

Jean Paul Degabriele1 and Kenneth G. Paterson2 *

Hewlett-Packard Laboratories, Bristol Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK. jeanpaul.degabriele@gmail.com ² Information Security Group, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK. Kenny.Paterson@rhul.ac.uk

Abstract. At Eurocrypt 2006, Paterson and Yau demonstrated how flaws in the Linux implementation of IPsec could be exploited to break encryption-only configurations of ESP, the IPsec encryption protocol. Their work highlighted the dangers of not using authenticated encryption in fielded systems, but did not constitute an attack on the actual IPsec standards themselves; in fact, the attacks of Paterson and Yau should be prevented by any standards-compliant IPsec implementation. In contrast, this paper describes new attacks which break any RFC-compliant implementation of IPsec making use of encryption-only ESP. The new attacks are both efficient and realistic: they are ciphertext-only and need only the capability to eavesdrop on ESP-encrypted traffic and to inject traffic into the network. The paper also reports our experiences in applying the attacks to a variety of implementations of IPsec, and reflects on what these experiences tell us about how security standards should be written so as to simplify the task of software developers.

Keywords: IPsec, integrity, encryption, ESP, standard.

What might go wrong?

- Bad choice of key size
 - Export-compatible key lengths
 - TI-x calculators RSA code signing keys

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	*	<pre>DH_anon_EXPORT</pre>	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

Export-Compatible Key Length



Source: http://www.copacobana.org

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 <u>message</u> 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 <u>website</u>.

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 <u>recent news</u>) Complete programming freedom has finally been achieved on the TI-83 Plus!

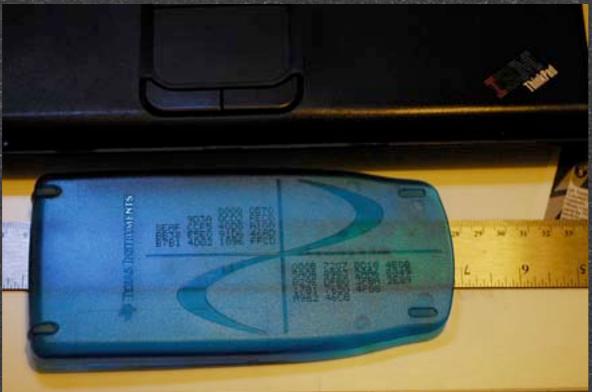
Update: Benjamin has posted additional details on the United-TI forum thread.

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

8

FloppusMaximus @

#11

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

What might go wrong?

- Bad use of cryptography
 - M\$ Lan Manager Hash
 - Incorrect use of RSA
 - Encrypt a symmetrical key without padding and with a small public exponent 8-))
 - Sign without any message preprocessing

What might go wrong?

And the list of horrors is still not at its end !!!

Fortunately...

- Correctly designed and properly analyzed primitives do exist:
 - Block cipher: AES (NIST FIPS 192)
 - Hash function: SHA-2 (NIST FIPS 180-2)
 - Public-key encryption and signature: RSA-OAEP and RSA-PSS (PKCS #1, v2.1)
 - Key-agreement protocol: ECDH (NIST SP800-56A)
 - Security protocol: TLS (RFC 5246)

So, has the lambda developer needing crypto any chance to stand on the safe side?

Well...

Side-Channel Attacks

Black-Box Adversaries

This the usual definition of an adversary for (theoretical) 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



Grey-Box Adversaries

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

side-

Timings channel information

Physical leakage

Faults

«tell»





White-Box Adversaries

- Adversaries that most 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

Embedded Platforms

- Tiny/small objects:
 - USB dongles
 - Smartcards
 - Chips
 - Embedded PCs
 - 8











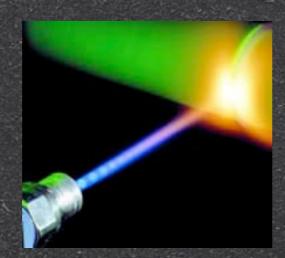
Embedded Platforms

- Main characteristics of embedded platforms for an attacker
 - Not remote, local
 - Rather slow clocks
 - Usually not very expensive

Side-Channel Attacks



- Timing
- Physical Leakage
- Faults





Timing Attacks

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

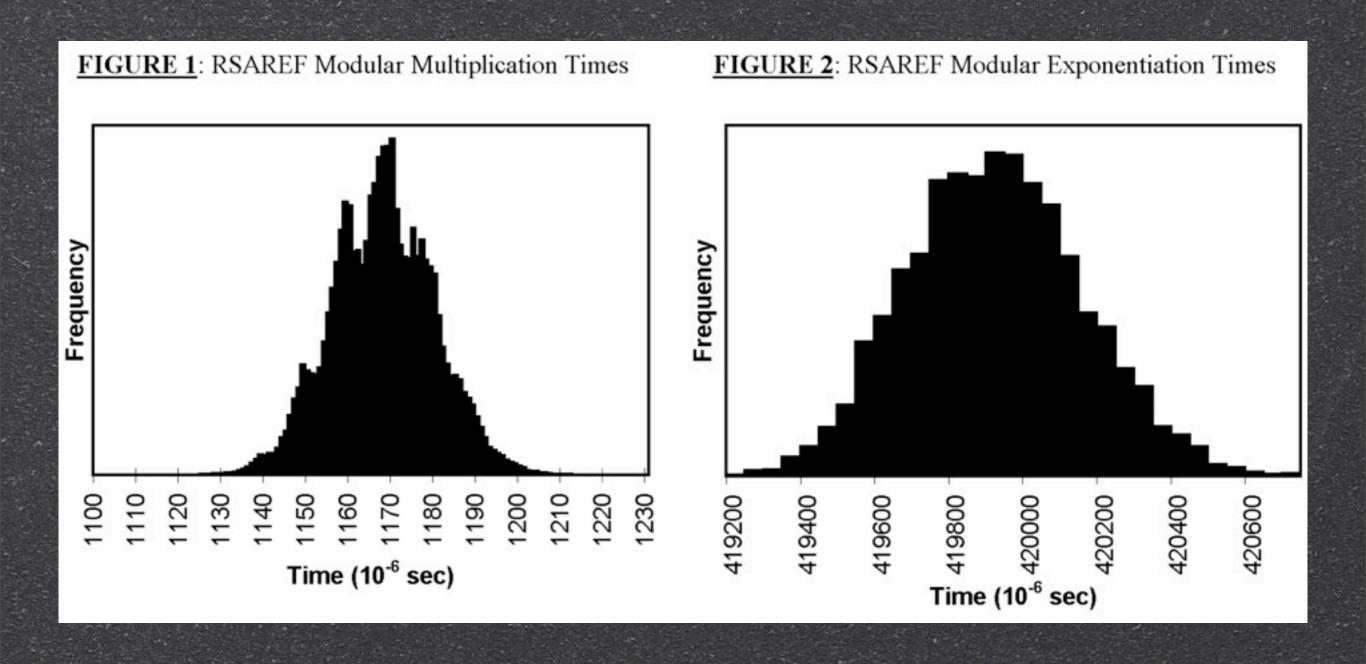
Paul C. Kocher

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

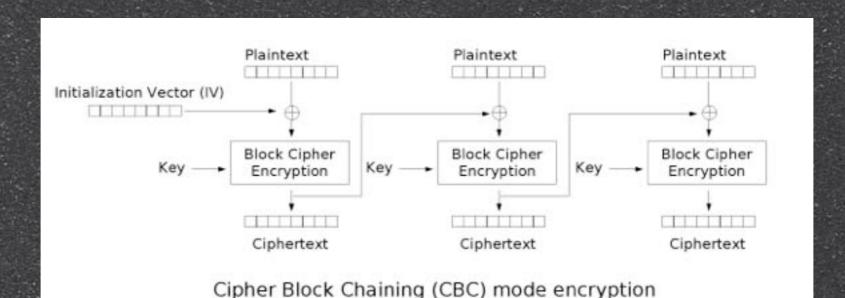
Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Again to vulnerable system, the attack is computationally inexperient and often requires only known depretext. 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.

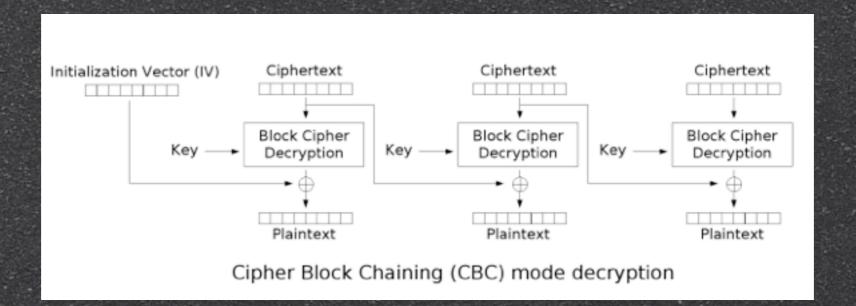
Keywords: timing attack, cryptanalysis, RSA, Diffie-Hellman, DSS.

Timing Attacks



Timing Attacks





- Encryption in CBC mode requires that the data have a length which is a multiple of the underlying block cipher block size.
 - AES-CBC: multiple of 16 bytes
 - TDES-CBC: multiple of 8 bytes

- 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 07 07 07 07 07
 - Missing 0 bytes: pad with 08 08 08 08 08 08

Problem if the padding checking routine is not time-constant:

Password Interception in a SSL/TLS Channel

Brice Canvel¹, Alain Hiltgen², Serge Vaudenay¹, and Martin Vuagnoux³

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

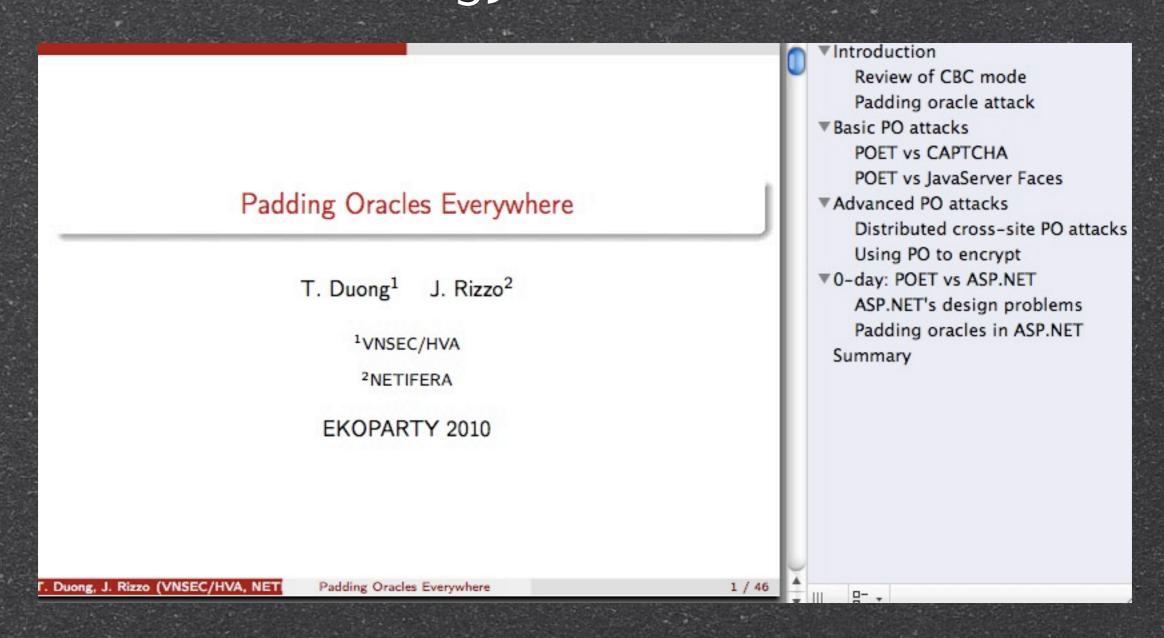
Abstract. Simple password authentication is often used e.g. from an email software application to a remote IMAP server. This is frequently done in a protected peer-to-peer tunnel, e.g. by SSL/TLS.

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.

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



Cache attacks:

Cache Attacks and Countermeasures: the Case of AES

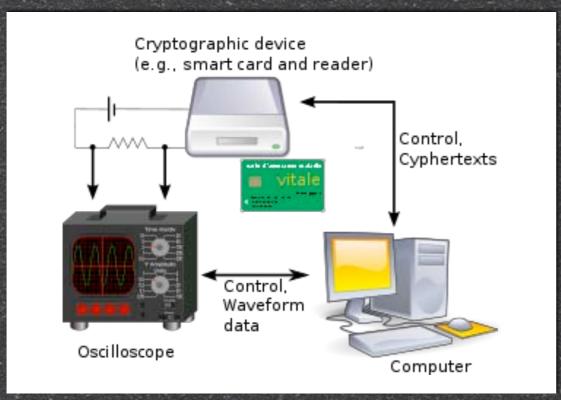
(Extended Version) revised 2005-11-20

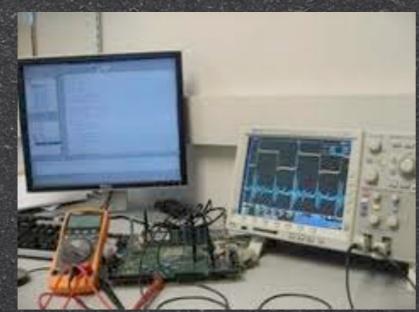
Dag Arne Osvik¹, Adi Shamir² and Eran Tromer²

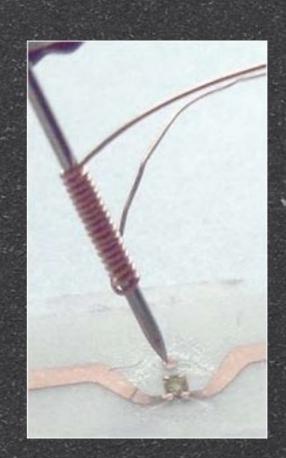
¹ dag.arne@osvik.no
² Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel {adi.shamir, eran.tromer}@weizmann.ac.il

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.

- 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!







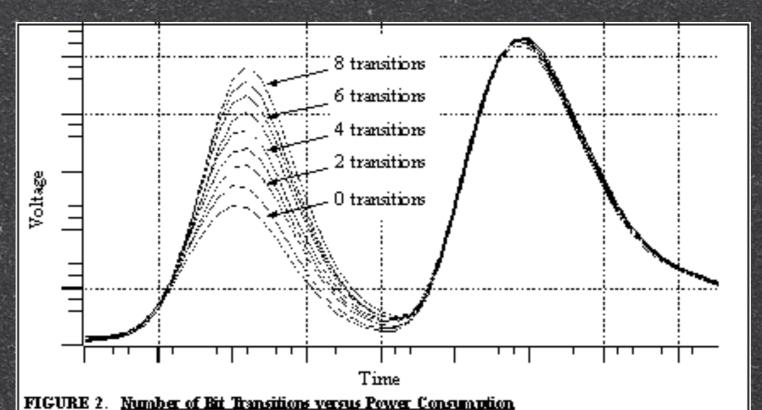
Differential Power Analysis

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



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 suggesting the power signals across 500 complex in order to

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:

cmp \$0x0, %ebx jne 0x64FE89A1

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

Attacks based on Faults Glitch Attacks

Design Principles for Tamper-Resistant Smartcard Processors

Oliver Kömmerling

Markus G. Kuhn

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

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

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.

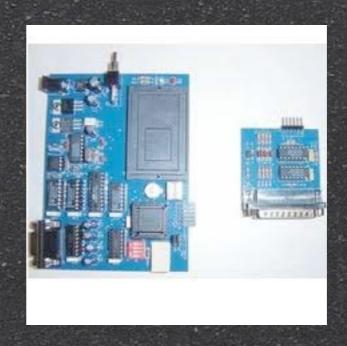
We are currently aware of three techniques for creating fairly reliable malfunctions that affect only a very small number of machine cycles in smartcard processors: clock signal transients, power supply transients, and external electrical field transients.

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







So, what about OpenSSL and Sisters?

OpenSSL and sisters

- Several general-purpose open-source cryptographic libraries do exist (nonexhaustive list):
 - OpenSSL
 - libgcrypt
 - Mozilla NSS
 - libtomcrypt

- NaCl
- Botan
- Crypto++
- cryptlib

OpenSSL and sisters

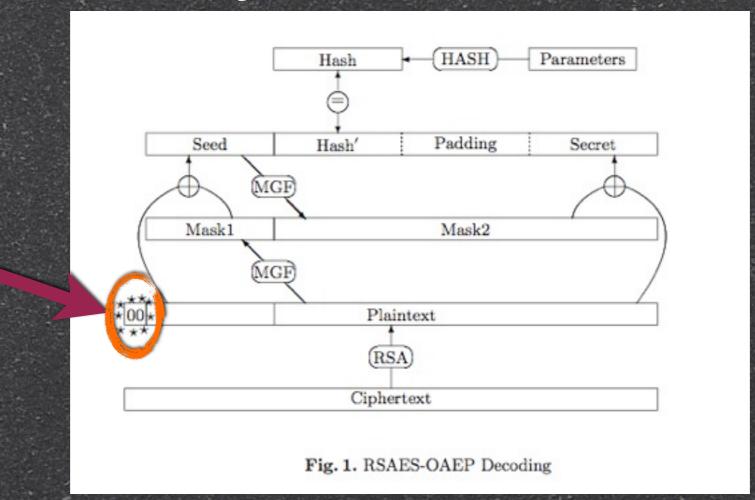
- Question I'd like to (partially)
 answer right now:
 - Blow secure are general-purpose opensource cryptographic libraries ?

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)

- 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

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



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

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]

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 */
```

```
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 */
```

0ut 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.

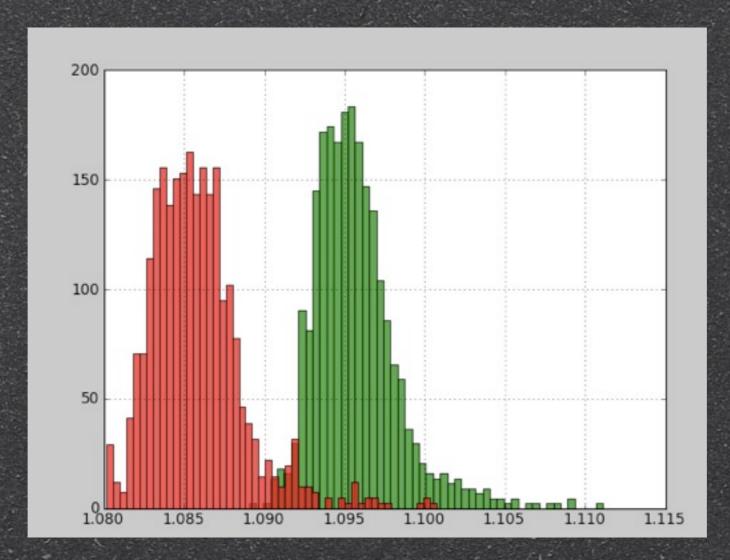
- Is that time-constant ?
- Time to compute 1'048'576 checks on my MacBook Pro:

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

Distribution of 1000 independent measures of 104'858 checks



- Is OpenSSL broken (with respect to Manger's attack)?
 - 0n 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...

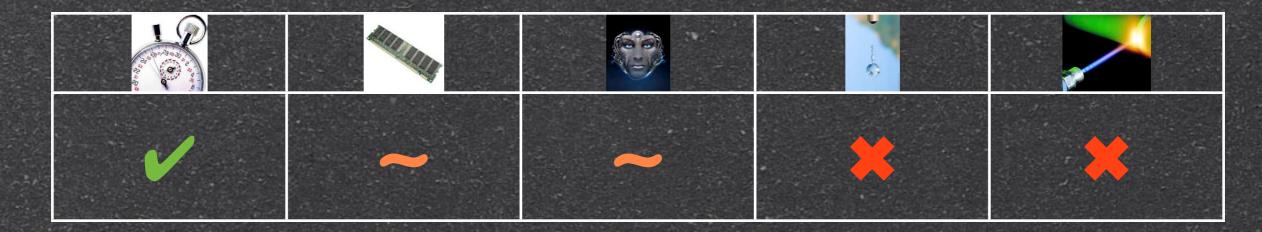
- Is OpenSSL broken (with respect to Manger's attack)?
 - 0n embedded platforms:
 - SYES, DEFINITIVELY !!
 - Clock-cycle accurate measurement is possible.
 - If time-constant, use the power trace of the execution.

Legend

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	Classical timing attacks				
	Cache attacks				
	Oracle attacks				
and C	Leakage attacks				
	Fault attacks				
/	Serious care				
~	Some care, but not always/properly				
	No care at all				

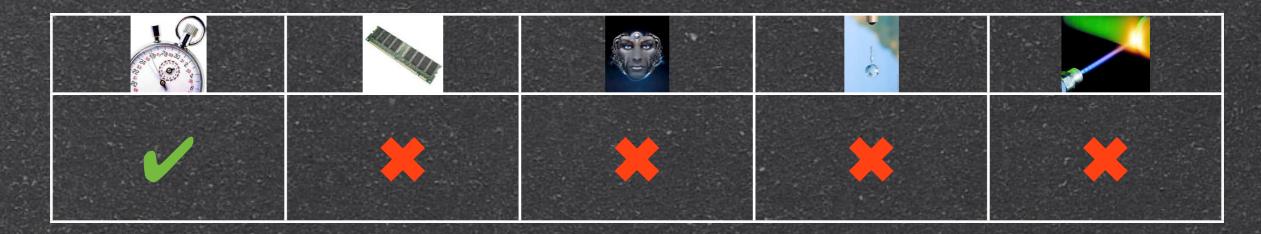
OpenSSL

- OpenSSL (http://www.openssl.org)
 - Most widely deployed general-purpose crypto library
 - Has an excellent reputation



libgcrypt

- libgcrypt (http://www.gnupg.org)
 - Written by the developers of the GnuPG application



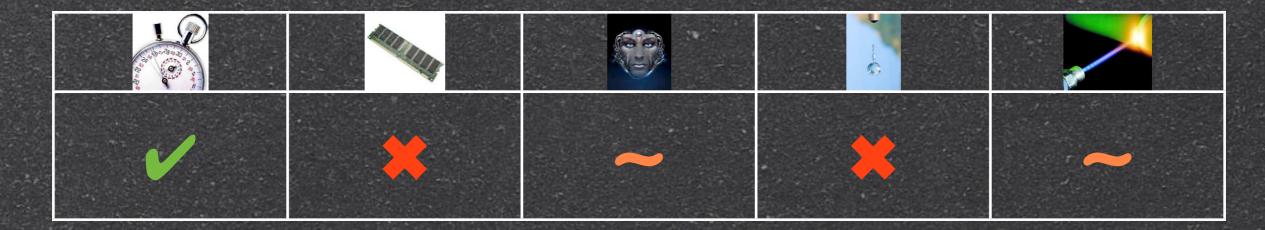
libtomcrypt

- ibtomcrypt (http://www.libtom.org)
 - Written by an enthusiastic teenager
 - Focus on speed...



Mozilla NSS

- NSS (http://www.mozilla.org)
 - Maintained by the Mozilla foundation

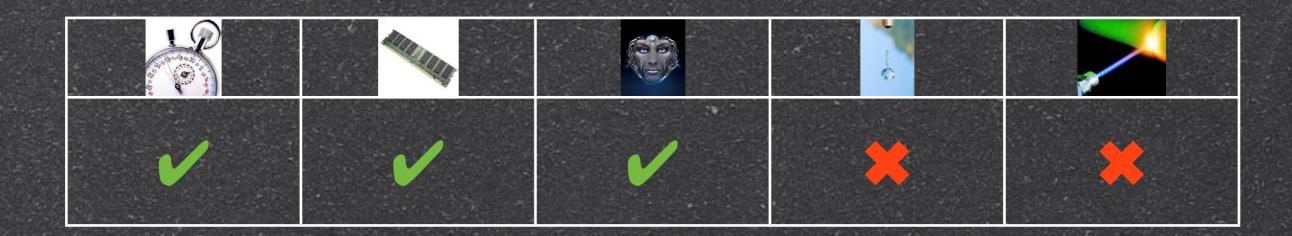


NaCl

- NaCl (http://nacl.cace-project.eu)
 - Written by cryptographers in the CACE European project
 - Features
 - High-speed implementations
 - No data-dependent branches

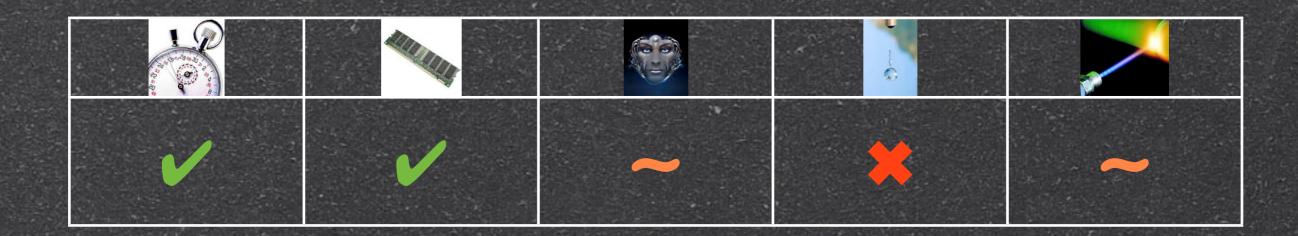
NaCl

Unfortunately, only a small set of exotic (though highly secure) primitives is supported



Botan

- Botan (http://botan.randombit.net)
 - Written in C++

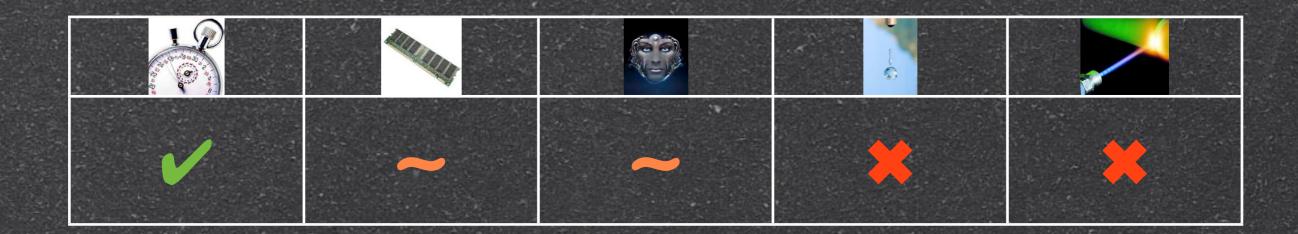


Botan

```
// 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++

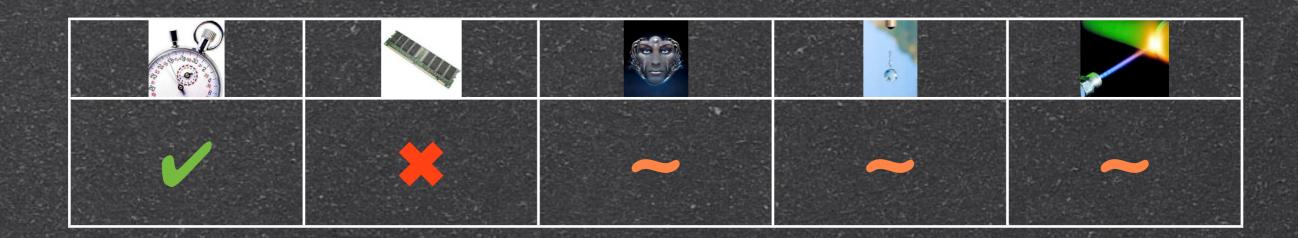
- Crypto++ (http://www.cryptopp.com)
 - Project maintained by Wei Dai
 - Conform to FIPS 140-2 level 1



Crypto++

cryptlib

- cryptlib (http://www.cs.auckland.ac.nz/
 ~pgut001/cryptlib)
 - Written by Peter Gutman



				Control of the contro	
0penSSL	'	~	~		
libgcrypt	/				
libtomcrypt					
NSS	~				~
NaCl	•	~	V		*
Botan	V	/	~		•
Crypto++	/	~	~		
cryptlib	/		~	~	~

- Although we have good general-purpose cryptography open-source libraries, they are not, or not completely protected against
 - oracle attacks,
 - leakage attacks,
 - and fault attacks.

- These observations have not an extreme impact on their security when their are used on production servers or desktop computers.
- But using current open-source cryptography on embedded platforms is very, very risky!

An efficient and properly secured open-source general-purpose cryptography library has still to be written!

Contact Information

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