Deliberately Un-Dependable Applications: the Role of Dependability Metrics in Fault-Based Cryptanalysis

Alfonso De Gregorio

C & A S.r.l.

www.com-and.com

8th August 2003

K.U.Leuven, COSIC-ESAT,

Kasteelpark Arenberg 10,

3001 Leuven-Heverlee

Agenda

- Introduction
  - Fault Attacks
  - The Model
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

Introduction

"Why erect majestic walls if comfortable underpasses will always remain wide open?"

Silvio Micali and Leonid Reyzin

Physically Observable Cryptography
<http://eprint.iacr.org/2003/120.pdf>

Introduction:
Feasibility of Physical Attacks

Essentially related to the way the attacker can interact with the target device and to the respective "observation" techniques available to the opponent:

- *Local Scenario:* Unsupervised physical access to one or more instances of cryptographic equipment:
  - Analysis of Sounds Given off by Rotor Machines
  - Timing Attacks
  - Power Analysis
  - Electromagnetic Emissions Analysis
  - Fault Attacks (injectable faults – e.g., optically)
- *Black-box Scenario*: Access to a remote device through a crypto protocol:
  - Fault Attacks (random occurring faults)

Fault Attacks: How faults interacts with security problems

Fact 1: *Active faults jeopardize security*

- Data errors: Erroneous cryptographic values enables an attacker *to expose the secret crypto key material*, without breaking the underlying algorithms
- Code errors: Single-bit control flow errors can completely *change the result of decision procedures* in many security context (e.g., authorization, authentication, rounds in iterated algorithms)

Fault Attacks: How faults interacts with security problems (ctd.)

Fact 2: *Their inherent availability threaten the very relevance of the schemes that are provably secure by a complexity-theoretic point of view*

Fault Attacks: exploiting them

Local Scenario / Induced Faults:

- Goal: to augment the occurrences of erroneous computations passed back to the user (i.e., fail-silent violations in f.t.s.)
- Note: inducing faults locally has been the only approach to increase the occurrences of fail-silent violations!
- Feasibility: the feasibility of inducing faults in diligently designed cryptographic modules is questioned by D.P. Maher and remains the *significant challenge* for the opponent:
  "Fault Injection Attacks, Tamper Resistance, and Hostile Reverse Engineering in Perspective", FC97
- *Partially addressed* in security standards: FIPS 140-2 (EFP and EFT), ISO 7816/{1,2} – ISO 7810, CC (FPT_PHP)

Fault Attacks: exploiting them (ctd)

Black-box model / Random occurring faults:

The probability of observing them is conditioned on the black-box dependability metrics and on the (standard) environmental conditions (e.g., SEU rates)

Fault Attacks: a first question

Is it possible to augment the occurrences of erroneous computations passed back to the user interface, in “diligently engineered” cryptographic modules, without inducing faults locally?

The Model

- A *cryptographic module* contains some cryptographic secret
- The module interacts with the outside world following a cryptographic protocol
- From time to time the module is affected by a random fault causing it to output erroneous values

The Model (ctd.)

- The cryptographic module has been designed to be evaluated by trusted third parties (e.g., evaluation/certification bodies) according to actual engineering standards at the highest assurance levels and the *correctness* of the implementations of its cryptographic algorithms has been formally verified

The Model (ctd.)

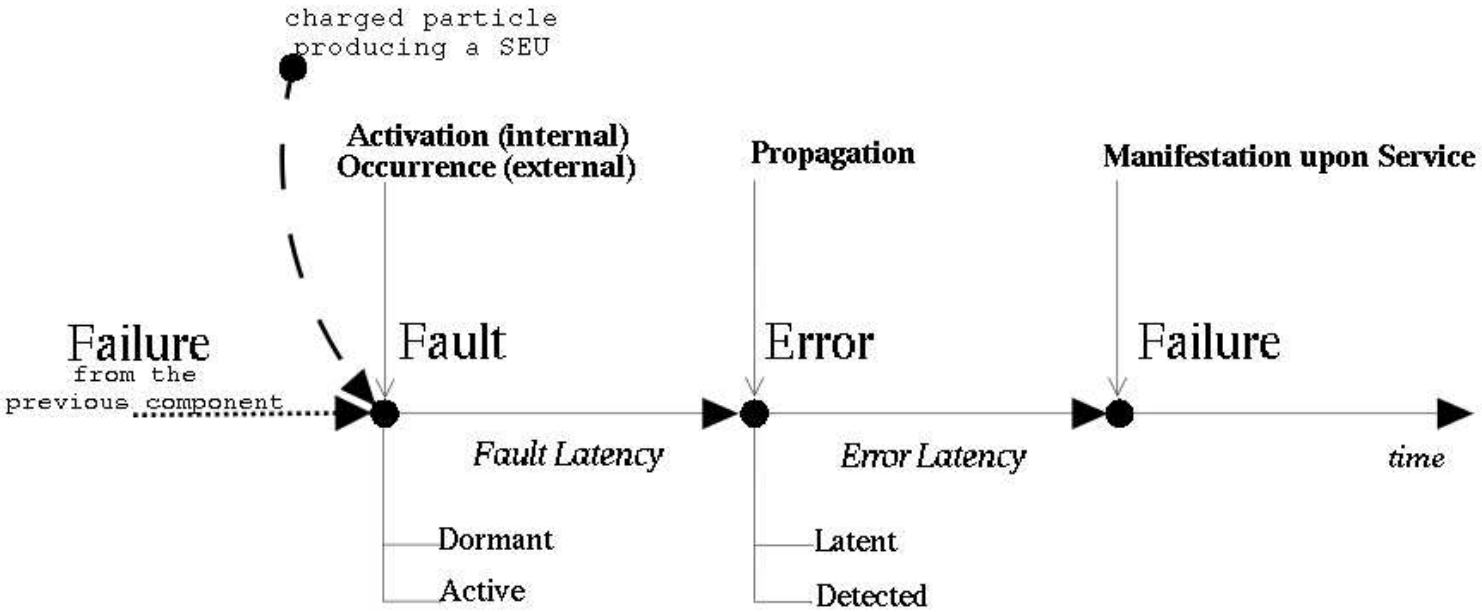
- *But it is still possible either to increase the occurrences of erroneous computations (resp. fail-silent violations for f.t.s.), or maximize the risks associated with the occurrence of faults by carefully augmenting the probability of observing naturally occurring and dormant faults at standard environmental conditions*

  Hence, to facilitate the feasibility of fault attacks

Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

Definitions and Background: Dependability Impairments



Definitions and Background: Transient Faults on Hardware

- Bit-flips (a form of transient faults) happen more in memory then in the processor (*Rad* Project)
- In RISC processors *(Furtado and Madeira):*
  - 60% cause data errors;
  - 30% cause code errors;
- In CISC processors:
  - 18% cause data errors;
  - 77% cause code errors;
- 20% of software failures are hw related (*Iyer*)

Definitions and Background: Design Faults on Hardware

Reported Design Faults in x86 µPs

µPs	Number of Reported Design Faults	
386 A1 step	28	
386 B0 step	12	
386 B1 step	15	
386 D0 step	3	
386 "some versions"	19	
386 "all versions"	1	
Total for 386 family		78
486 "early versions"	6	
486 "some versions"	8	
486 A-B4 steps	3	
486 A-C0 steps	2	
486 "all versions"	2	
Total for 386 family		21
Pentium 60 and 66-Mhz	21	
Pentium 75, 90, 100-Mhz	42	
Total for Pentium (to Feb. 1995)		56

- Silber, Porras, and Lindell, "The Intel 80x86 Processor Architecture" IEEE C.S. Symposium on Research in Security and Privacy, May, 1995
- Donald MacKenzie, "Mechanizing Proof – Computing, Risk, Trust", MIT Press

Definitions and Background: Dormant Faults on Software
(an example dated 09 July 2003)

Outside our Model

- Cryptlib Cryptographic Toolkit:
  - On Sat 12 July the author, following an bug report dated 09 July, announce that due to a problem in the RSA implementation in some *rare occasions* the toolkit computes *erroneous RSA signatures*
  - Cryptlib was computing 0.3% of erroneous signatures
  - Its RSA implementation uses the Chinese Remainder Theorem
  - The bug was present also in OpenSSL (some time ago)

Definitions and Background: Dormant Faults on Software

(an example dated 09 July 2003 - ctd.)

Outside our Model

Incorrect

```
Int rsaDecrypt( CRYPT_INFO *cryptInfo,
   BYTE *buffer, int noBytes )

  {  /* ... snip ... */

/* computing:

 * p2 = ((C mod p) **exponent1) mod p;

 * q2 = ((C mod q) **exponent1) mod p;

 * ... */

 /* p2 = p2 - q2; if p2 < 0 then p2 =
   p2 + p */

  CK( BN_sub( p2, p2, q2 ) );

  if( p2->neg )

    CK( BN_add( p2, p2, p ) );

/* ... */
   }
```

Correct

```
int rsaDecrypt( CRYPT_INFO *cryptInfo,
   BYTE *buffer, int noBytes )

  {  /* ... snip ... */

/* p2 = p2 - q2; if p2 < 0 then p2 =
   p2 + p. In some extremely rare
   cases (q2 large, p2 small) we have
   to add p twice to get p2 positive
   */

   CK( BN_sub( p2, p2, q2 ) );
   while( p2->neg )
   {

     CK( BN_add( p2, p2, p ) );
       if( bnStatusError( bnStatus )
           return(\
           getBnStatus(bnStatus ) );

    }

/* ... */
   }
```

Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

DUDA

- A class of malware
- A twofold phenomenological cause of faults
- Absence of additional (malicious) logics
- Embedded at design time
- Aimed at facilitating fault attacks
- By maximizing either the probability of observing faults, or the risks associated to the occurrences/activation of them

DUDA (ctd.)

Attributes / Malware Classes	Trojan horse	Logic bomb	Trapdoor	Worm	Virus	DUDA
Phenomenological Cause	human-made	human-made	human-made	human-made	human-made	Nature and Human-made
Phase of System Life	operational	design	design	operational	operational	design
Absence of Additional Logics	No	No	No	No	No	Yes

Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics

- Dependability metrics has often been observed to correlate with workload characteristics. In particular with either:
  - Computational load of the system
    - Memory or CPU utilization
  - Or the characteristics of its application code:
    - Read/write ratio
    - Memory allocation schemes / Access patterns
- For instance, *failure rates* increase with workload (and mission times resp. decreases)

{De,In}creasing the Probability of Observing Faults: the case of memory

- The possible consequences of memory faults:

1. The (target) software is *not affected*
2. The fault affects the code part producing a crash
3. The fault affects the code part causing an erroneous state that may be (or not) corrected by the handling mechanism in software, if any.
4. The data part gets affected in a fatal way
5. The data part gets affected but not in a fatal way. The erroneous memory value(s) causes a “drift” in subsequent computations
6. The data part gets affected but *the error is masked*

{De,In}creasing the Probability of Observing Faults: the case of memory (ctd.)

A first instance of DUDA

- Some results from the reliability engineering community:
  - Workload characteristics have a major impact on memory reliability (Meyer, Wei)
  - *Different memory allocation schemes produce different increments in failure rates in the range of 32% to 53% (Bowen, Pradhan)*

{De,In}creasing the Probability of Observing Faults: the case of memory (ctd.)

A first instance of DUDA

- When a system runs at a low load
  - The programs are able to execute with the amount of dead blocks they requested to the “memory manager”
  - A considerable amount of errors do not cause failures because of the increased probability of hitting a dead block

{De,In}creasing the Probability of Observing Faults: the case of memory (ctd.)

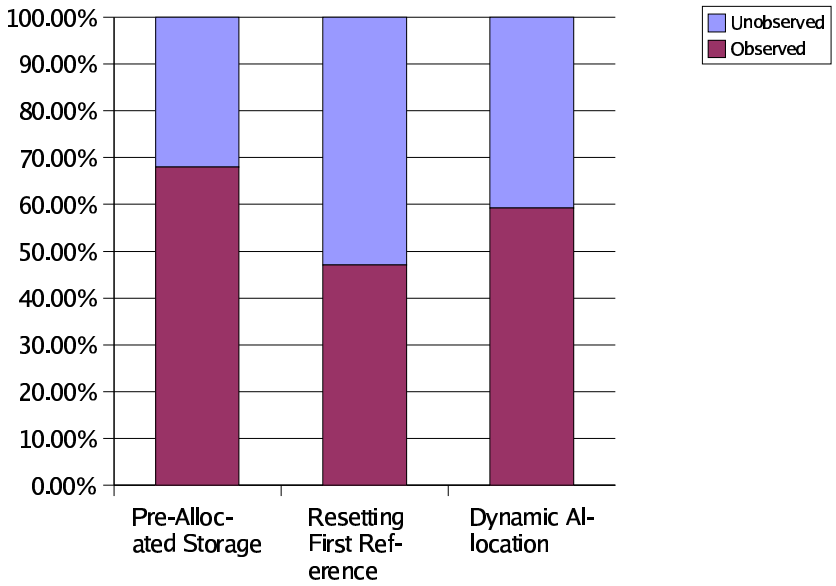
A first instance of DUDA

- As the workload increase:
  - The operating system becomes resource constrained and thus trims away many of the dead block leaving each job with a much higher percentage of live blocks
  - This increases the probability that a fault affects a live block and thus the observed failure rate increases

{De,In}creasing the Probability of Observing Faults: the case of memory (ctd.)

$$P(U \mid MemFail) = \frac{Pr[\#\text{ dead-block fault} \geq 1]\, Pr[\#\text{ "in-use" block faults} = 0]}{Pr[\#\text{ total-block faults >= } 1]}$$

P_u	Pre-Allocated Storage	Resetting First Reference	Dynamic Allocation
Average	0.320	0.530	0.408
Low	0.290	0.446	0.343
High	0.369	0.628	0.498



Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

The Impact on Security Policies: Selecting Key-Lifetimes

- **Definition 1**: Cryptographic Scheme Failure-Tolerance

  Informally: The maximum number $\mathcal{F}$ of erroneous computations that a cryptographic black-box implementing the cryptographic scheme $\mathcal{S}$ can output before the key-material get exposed by a fault attack

Cryptographic Scheme	CSFT	Author(s)	
Fiat-Shamir Id. Scheme (2^{-40} error bound)	~10	*Boneh, DeMillo and Lipton*	1996
RSA (1024 bit)	~1000	*Boneh, DeMillo and Lipton*	1996
Schnorr's Id. Protocol (2^{-40} error bound)	~10,000	*Boneh, DeMillo and Lipton*	1996
RSA+CRT	0	*Lenstra*	1997
AES-128 1st attack	49	*Giraud*	2002
AES-128 2nd attack	249	*Giraud*	2002

The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)

- Consider to compare three similar cryptographic device $\mathcal{D}1, \mathcal{D}2$ and $\mathcal{D}3$ implementing the cryptographic scheme $\mathcal{S}$
- $\mathcal{D}1$ = reference implementation, $\mathcal{D}2$ = modeled considering the workload with max. un-observability, $\mathcal{D}3$ = workload with min. un-observability
- They have more operational states. The service is given only while at a particular state denominated *service*.
- Each of them takes γ_d hours before entering the *service* state
- The time of occurrence of the failures in the system is considered to be a random variable with the corresponding distribution being exponential with two-variables with rates respectively equal to $\lambda 1$, $\lambda 2$, $\lambda 3$ and $\gamma = \gamma_d$

The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)

- The reliability of the system at time *t* is expressed as:

$$R_{sys} = e^{-\lambda (t - \gamma)}$$

- The system is considered to be functioning as long as the key material has not been exposed (i.e., as long as the number of failure is less than or equal to the CSFT value of the cryptographic scheme implemented)
- While providing service, the error probability must be kept less than or equal to ε

The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)

- Hence the *lifetime of the key-material* should NOT exceed the *exponential reliable life* (i.e., mission time) defined as:

$$KeyLifetime \leq t_R = \gamma - \frac{\ln\left(1 - \sqrt[F+1]{\epsilon}\right)}{\lambda}$$

where $1 - \sqrt[F+1]{\epsilon}$ is the desired reliability goal.

The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)

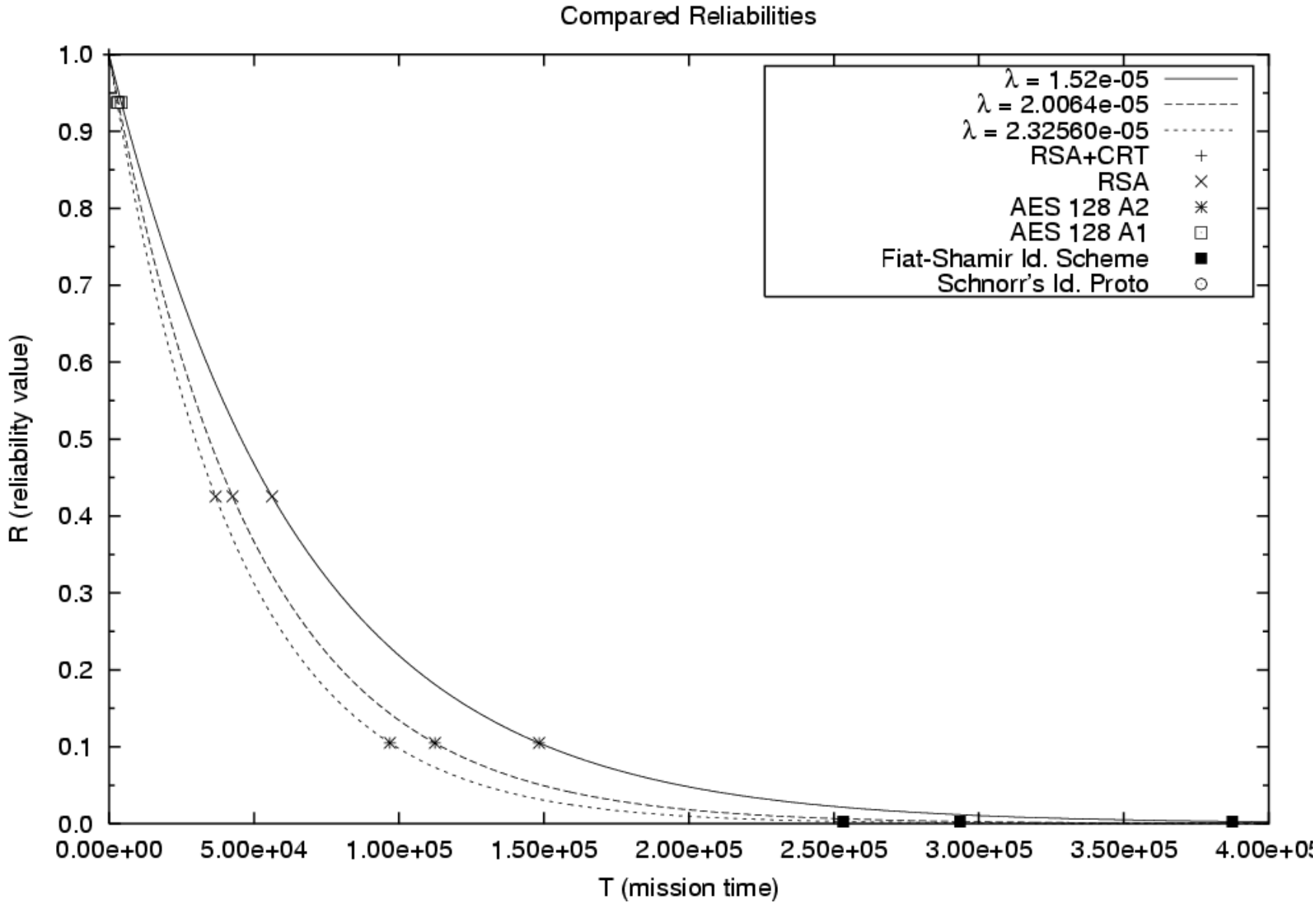
With $\varepsilon = 2e\text{-}40$, $\gamma = 10$ hours,

$\lambda_1 = 1.52e\text{-}05$, $\lambda_2 = 2.0064e\text{-}05$, $\lambda_3 = 2.32560e\text{-}05$

MTTF1 = 65800 MTTF2 = 49851 MTTF3 = 43010

	t1 in hours	t2	t3	Δ(t3-t2)	Δ%
Fiat-Shamir Id. Scheme (2e-40)	4.26E+003	3.23E+003	2.79E+003	441.5	13.68
RSA (1024)	2.37E+005	1.79E+005	1.55E+005	24621.74	13.72
Schnorr's Id. Proto (2e-40)	3.87E+005	2.94E+005	2.53E+005	40288.25	13.73
RSA + CRT	**1.00E+001**	**1.00E+001**	**1.00E+001**	**0**	**0**
AES 128 Attack 1	**5.62E+004**	**4.26E+004**	**3.67E+004**	**5843.02**	**13.72**
AES 128 Attack 2	**1.48E+005**	**1.12E+005**	**9.69E+004**	**15419.52**	**13.72**

The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)



The Impact on Security Policies: Selecting Key-Lifetimes (ctd.)

- The importance of dependability metrics should not be overlooked. By exploiting them (e.g., increasing the *f.r.* / decreasing the MTTF) it's possible to facilitate fault attacks.
- Reliability modeling gives us a boundary for the key-lifetimes of cryptographic schemes implemented in real devices: the system *reliable life*
- RSA+CRT cannot be securely used in today systems with mid-level failure rates, while requiring an unreliability less than or equal to a negligible probability error

Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

The Impact on Standards

- Claim 1: The presented class of attacks pass unnoticed to the assessment techniques used in today security standards
- Claim 2: It's necessary to support today standards with additional and definite dependability requirements

The Impact on Standards: FIPS 140-2

Security Level 4 Requirements

- Environmental failure protection (EFP):
  protection against *unusual environmental conditions or fluctuations (accidental or induced) outside the module's normal operating range* that can compromise the security of the module – with *temperature* and *voltage* mention explicitly
- Environmental failure testing (EFT): combination of analysis, simulation and testing of cryptographic module to provide reasonable assurance that no compromise of the security is possible due to environmental conditions *fluctuations*
- **FIPS 140-2 do not address augmented failure rates at standard environmental conditions**

The Impact on Standards: FIPS 140-2 (ctd.)

The Problem of the Correctness Notion: or How Badly a "Correct" Implementation Can Behave?

- Informal and formal proofs of correspondence between different abstraction levels:
  - A formal model for the rules and characteristics of the cryptographic module security policy, using a formal specification language
  - An informal proof of the correspondence between the formal model and the functional specifications
  - For each component, function and procedure a description of its behavior using {pre,post}conditions
  - Informal proof of the correspondence between the design and the functional specification

The Impact on Standards: FIPS 140-2 (ctd.)

The Problem of the Correctness Notion: or How Badly a "Correct" Implementation Can Behave?

- However, using DUDAs it is possible to add malicious behaviors without the need for additional logics
- A formal proof of correspondence between the design and the functional specification of a device **is not sufficient** to rule out the presence of additional and unexpected malicious behaviors.
- Hence, relying parties are unable to build a trust relationship with cryptographic module fully compliant with today standard requirements.

Overview of the Talk

- Introduction
- Definitions and Background
- Deliberately Un-Dependable Applications
- Facilitating the Practicality of Fault Attacks by Exploiting Workload Characteristics
- The Impact on Security Policies
- The Impact on Standards
- Conclusions

Conclusions

- Inducing fault locally is not the only way to increase the occurrences of erroneous computations in security devices

- Augmenting the probability of observing natural occurring and dormant faults is an alternative

Conclusions (ctd.)

- A new class of malware has been introduced
- Its nature is subtle and unique
  - The attack paradigm takes advantage of faults with a twofold phenomenological cause
  - And enables manufacturers to add malicious behaviors without additional logics
- It gives us an attack model against security modules
- A first instance of DUDA has been presented. It is show how dependability metrics can be undermined by exploiting workload characteristics

Conclusions (ctd.)

- Its possible to use any cryptographic scheme without being afraid of fault attacks already known in literature, *if the reliability of the cryptographic module is carefully modeled*.
  - The exponential reliable life give us a boundary for key-lifetimes
- RSA+CRT cannot be securely used in today systems with mid-level failure rates, while requiring an unreliability less than or equal to a standard negligible probability error for cryptographic context (2e-40)

Conclusions (ctd.)

- DUDA attacks passes unnoticed to the assessment techniques used to evaluate today cryptographic modules at the highest “security” levels

- More importantly:
  There is the need to support today standards with more definite dependability requirements.

Conclusions: Further Works

- Complete a first taxonomy of DUDA based attacks on security systems
- Provide a generalized framework to model the reliability of complex cryptographic infrastructures.
  - It will guide us in recommending key-lifetimes in presence of faults.
- Complete the analysis on the impact on standards

Thanks!

Any question?

Alfonso.DeGregorio@esat.kuleuven.ac.be
adg@com-and.com