Cryptanalysis vs. Reality

Jean-Philippe Aumasson

`http://131002.net` `@aumasson`

Cryptanalysis is the study of methods for obtaining the meaning of encrypted information without access to the secret information that is normally required to do so. *Wikipedia*

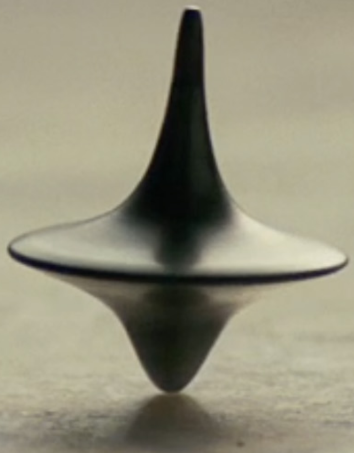
Cryptanalysis is the study of methods for obtaining the meaning of encrypted information without access to the secret information that is normally required to do so. *Wikipedia*

> The fundamental goal of a cryptanalyst is to violate one or several security notions for algorithms that claim, implicitly or explicitly, to satisfy these security notions.
>
> *Antoine Joux, Algorithmic Cryptanalysis*

Reality noun (pl. realities)

1. the state of things as they actually exist, as opposed to an idealistic or notional idea of them.
2. a thing that is actually experienced or seen.
3. the quality of being lifelike.
4. the state or quality of having existence or substance.

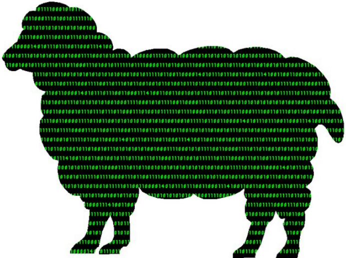
Compact Oxford English Dictionary



Cryptanalysis relies on an **ATTACKER MODEL**
= *assumptions on what the attacker can and cannot do*

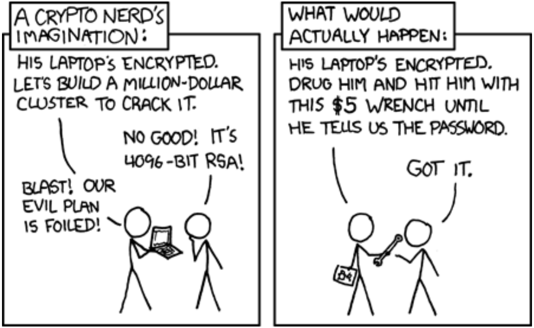
All models are in **simulacra**, that is, simplified reflections of reality, but, despite their inherent falsity, they are *nevertheless extremely useful*

G. Box, N. Draper, Empirical Model-Building and Response Surfaces

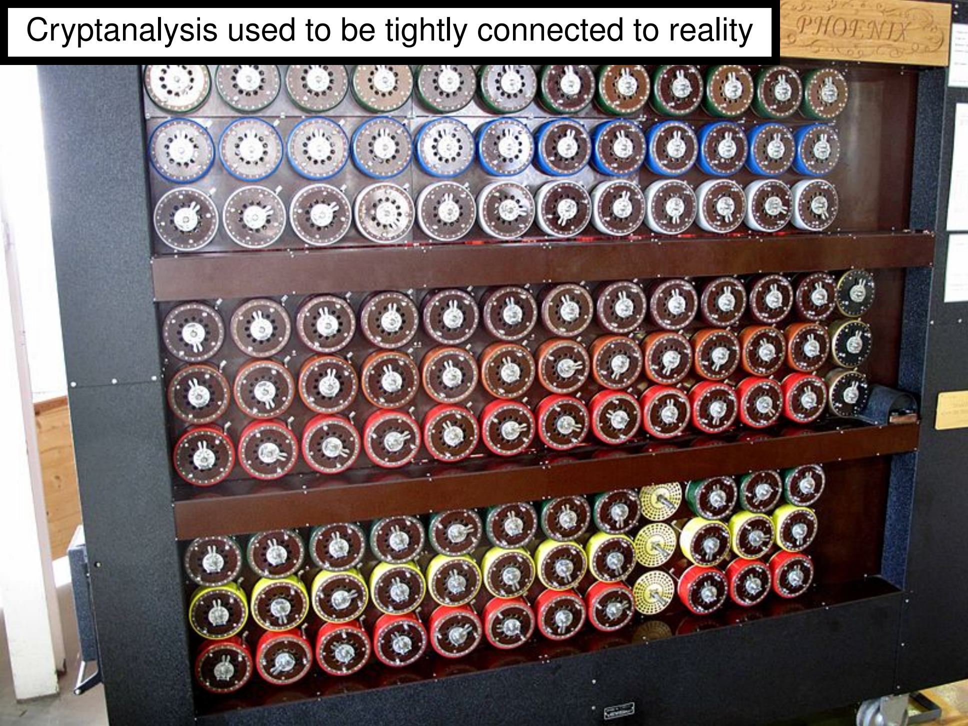


Cryptanalysis usually excludes methods of attack that do not primarily target weaknesses in the actual cryptography, such as bribery, physical coercion, burglary, keystroke logging, and social engineering, although these types of attack are an important concern and are often more effective

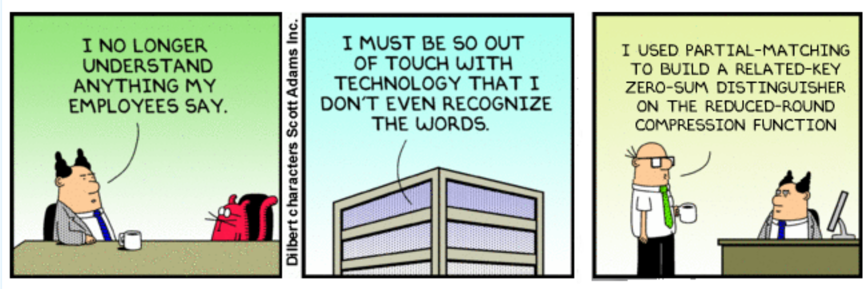
Wikipedia



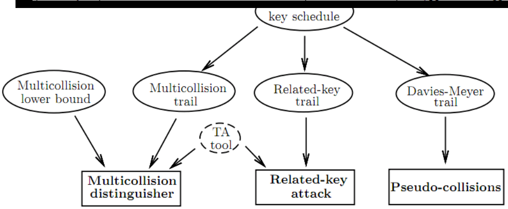
Cryptanalysis used to be tightly connected to reality



Times have changed



SHA3 Round	Best Known Analysis	Rounds / total	Previous Time	Previous Memory	Ref.	This paper Time	This paper Memory
Final	semi-free-start coll.	16 / 42	2^{190}	2^{104}	[16]	$\mathbf{2^{97}}$	$\mathbf{2^{97}}$
	semi-free-start near coll.	22 / 42	2^{168}	$2^{143.70}$	[16]	$\mathbf{2^{96}}$	$\mathbf{2^{96}}$
Final*	(compr. function property)	10 / 10	2^{192}	2^{64}	[15]	$\mathbf{2^{182}}$	2^{64}
	(internal permutation dist.)	10 / 10	2^{192}	2^{64}	[15]	$\mathbf{2^{175}}$	2^{64}
	(compr. function property)	11 / 14	2^{640}	2^{64}	[15]	$\mathbf{2^{630}}$	2^{64}
2^{nd}	internal permutation dist.	8 / 8	2^{182}	2^{37}	[17]	$\mathbf{2^{151}}$	2^{67}



6.2 Related-Key I

Like in our previous a
text that vanish unti
differentials). Then, w
the cipher, i.e., between the 16-th and 17-th rounds. Our differential trail for E^{β}
has probability $p = 2^{-86}$, and the one for E^{γ} has probability 2^{-113}, leading to a boomerang distinguisher on 34 rounds requiring about $(pq)^{-2} = 2^{398}$ trials. The trails used are described in detail in Appendix D. Note that for the second part, MSB differences are set in the key words k_2 and k_3, and in the tweak words t_0 and t_1 (thus giving no difference in the seventh subkey).

6.3 Known-Related-Key Distinguishers

Although the standard notion of distinguisher requires a secret (key), the notion of *known-key distinguisher* [22] is also relevant to set apart a block cipher from

Hardware > Security

AES encryption is cracked

Researchers find a weakness in the algorithm

By Dave Neal

Wed Aug 17 2011, 11:55

January 11, 2010, 4:57PM

A Second GSM Cipher Falls

by Dennis Fisher

Follow @DennisF

Sh

4 Comments

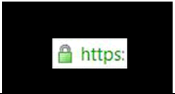
A group of cryptographers has developed a new attack that has broken Kasumi, the encryption algorithm used to secure traffic on 3G GSM wireless networks. The technique enables them to recover a full key by using a tactic known as a related-key attack, but experts say it is not the

SECURITY

Hackers Crack Internet Encryption: Should You Be Worried?

By Alex Wawro, PCWorld

Data encryption is the cornerstone of Internet security. Every time you log into your email account or sign into an online retailer like Amazon, chances are that your browser is establishing a secure connection to the server using an encryption technology called TLS (Transport Layer Security).



Hardware > Security

AES encryption is cracked

Researchers find a weakness in the algorithm

By Da

Wed A

Broken in a model does not imply broken in reality!

Sh

s

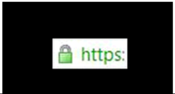
A group of cryptographers has developed a new attack that has broken Kasumi, the encryption algorithm used to secure traffic on 3G GSM wireless networks. The technique enables them to recover a full key by using a tactic known as a related-key attack, but experts say it is not the

SECURITY

Hackers Crack Internet Encryption: Should You Be Worried?

By Alex Wawro, PCWorld

Data encryption is the cornerstone of Internet security. Every time you log into your email account or sign into an online retailer like Amazon, chances are that your browser is establishing a secure connection to the server using an encryption technology called TLS (Transport Layer Security).



Models’ language overlaps with real-world language: “attacks”, “broken” have different meanings

Have we lost connection with reality?

Cryptography is usually bypassed. I am not aware of any major world-class security system employing cryptography in which the hackers penetrated the system by actually going through the cryptanalysis. (...) Usually there are much simpler ways of penetrating the security system.

Adi Shamir, Turing Award lecture, 2002

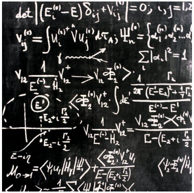
Is cryptanalysis relevant at all?

Part 1: Physical attacks

- Bypass and misuse
- Side-channel attacks
- Leakage-resilient crypto

Part 2: Algorithmic attacks

- State-of-the-ciphers
- Why attacks aren't attacks
- Cognitive biases
- An attack that works
- What about AES?



Part 1: Physical attacks

- Bypass and misuse
- Side-channel attacks
- Leakage-resilient crypto

HTTPS server authentication with **2048-bit RSA**
$\approx$ 100-bit security [http://www.keylength.com/]

$\approx 2^{100} \approx 10^{30}$ ops to break RSA by factorization

HTTPS server authentication with **2048-bit RSA**
$\approx$ 100-bit security [`http://www.keylength.com/`]

$\approx 2^{100} \approx 10^{30}$ ops to break RSA by factorization

$\approx 2^{33}$ using a **quantum computer** implementing Shor's algorithm

HTTPS server authentication with **2048-bit RSA**
$\approx$ 100-bit security [`http://www.keylength.com/`]

$\approx 2^{100} \approx 10^{30}$ ops to break RSA by factorization

$\approx 2^{33}$ using a **quantum computer**
implementing Shor's algorithm

$\approx$ 0 by compromising a trusted CA. . .

```
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        05:e2:e6:a4:cd:09:ea:54:d6:65:b0:75:fe:22:a2:56
    Signature Algorithm: sha1WithRSAEncryption
    Issuer:
        emailAddress              = info@diginotar.nl
        commonName                = DigiNotar Public CA 2025
        organizationName          = DigiNotar
        countryName               = NL
    Validity
        Not Before: Jul 10 19:06:30 2011 GMT
        Not After : Jul  9 19:06:30 2013 GMT
    Subject:
        commonName                = *.google.com
        serialNumber              = PK000229200002
        localityName              = Mountain View
        organizationName          = Google Inc
```

Researchers Crack HD Content Protection System

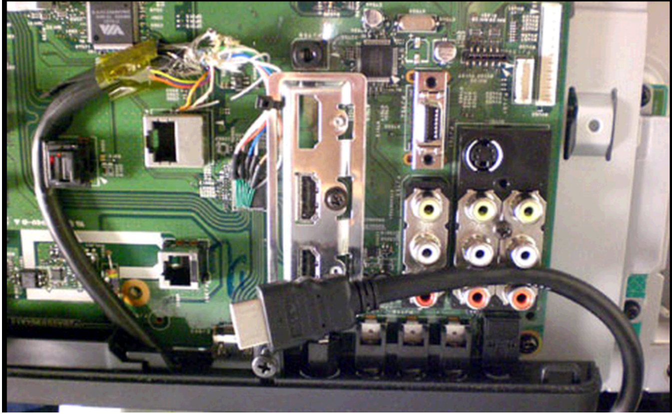
November 25, 2011 By Ethical Hacker Leave a Comment

(LiveHacking.Com) - Security researchers have broken the High-bandwidth Digital Content Protection (HDCP) system used on HD devices (such as Blu-ray) with HDMI ports to protect digital video sent to TVs and monitors against unauthorized copying.

Using a man-in-the-middle (or in this case a computer board in the middle), Prof. Dr.-Ing Tim Güneysu of the Secure Hardware Group at Germany's Ruhr University of Bochum, has found a way to connect any non-compliant monitor (which would include devices able to record the video) to a HDCP protected video source.

TV hack bypasses HDCP

posted Oct 1st 2009 1:00pm by Phil Burgess
filed under: home entertainment hacks, video hacks



ECDSA signing with a constant
instead of a random number
to find SONY PS3's private key

ECDSA signing with a constant instead of a random number
to find SONY PS3's private key

RC4 stream cipher with part of the key public and predictable in WiFi's WEP protection)

ECDSA signing with a constant
instead of a random number
to find SONY PS3's private key

RC4 stream cipher with part of the key public and
predictable in WiFi's WEP protection)

TEA block cipher in hashing mode
to perform boot code authentication
Equivalent keys = collisions = break

Remote side-channel attacks

Breaking the "secure" AES of **OpenSSL 0.9.8n**:

Cache Games – Bringing Access-Based Cache Attacks on AES to Practice

Endre Bangerter
Bern University of Applied Sciences
endre.bangerter@bfh.ch

David Gullasch
Bern University of Applied Sciences,
Dreamlab Technologies
david.gullasch@bfh.ch

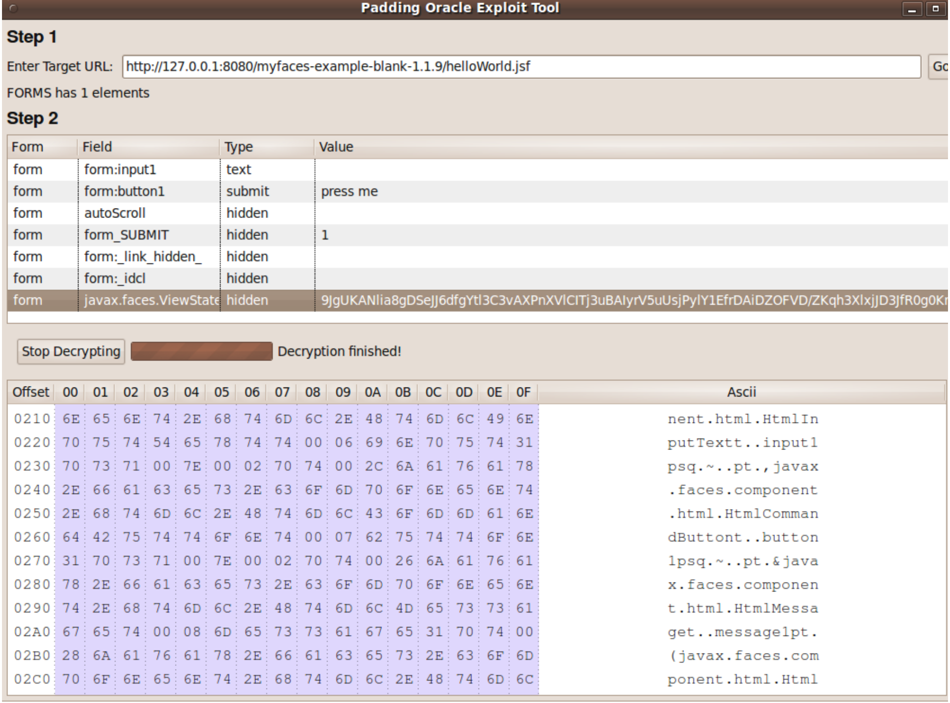
Stephan Krenn
Bern University of Applied Sciences,
University of Fribourg
stephan.krenn@bfh.ch

Breaking AES on **ARM9**:

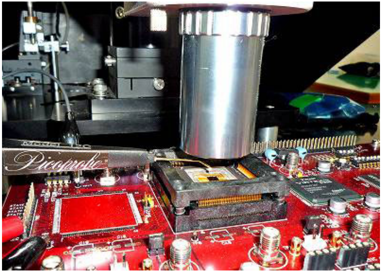
Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs

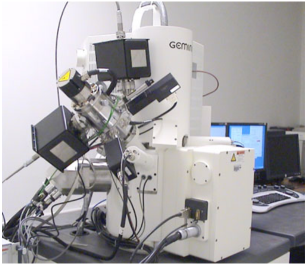
Andrey Bogdanov[1], Thomas Eisenbarth[2], Christof Paar[2], Malte Wienecke[2]

[1] Dept. ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Belgium
`andrey.bogdanov@esat.kuleuven.be`
[2] Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany
{`thomas.eisenbarth, christof.paar, malte.wienecke`}`@rub.de`


Padding Oracle Exploit Tool
Step 1
Enter Target URL: http://127.0.0.1:8080/myfaces-example-blank-1.1.9/helloWorld.jsf
FORMS has 1 elements
Step 2
Form Field Type Value
form form:input1 text
form form:button1 submit press me
form autoScroll hidden
form form_SUBMIT hidden 1
form form:_link_hidden_ hidden
form form:_idcl hidden
form javax.faces.ViewState hidden 9JgUKANlia8gDSeJJ6dfgYtl3C3vAXPnXVlCITj3uBAIyrV5uUsjPylY1EfrDAiDZOFVD/ZKqh3XlxjJD3JfR0g0Kr
Stop Decrypting
Decryption finished!
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Ascii
0210 6E 65 6E 74 2E 68 74 6D 6C 2E 48 74 6D 6C 49 6E nent.html.HtmlIn
0220 70 75 74 54 65 78 74 74 00 06 69 6E 70 75 74 31 putTextt..input1
0230 70 73 71 00 7E 00 02 70 74 00 2C 6A 61 76 61 78 psq.~..pt.,javax
0240 2E 66 61 63 65 73 2E 63 6F 6D 70 6F 6E 65 6E 74 .faces.component
0250 2E 68 74 6D 6C 2E 48 74 6D 6C 43 6F 6D 6D 61 6E .html.HtmlComman
0260 64 42 75 74 74 6F 6E 74 00 07 62 75 74 74 6F 6E dButtont..button
0270 31 70 73 71 00 7E 00 02 70 74 00 26 6A 61 76 61 1psq.~..pt.&java
0280 78 2E 66 61 63 65 73 2E 63 6F 6D 70 6F 6E 65 6E x.faces.componen
0290 74 2E 68 74 6D 6C 2E 48 74 6D 6C 4D 65 73 73 61 t.html.HtmlMessa
02A0 67 65 74 00 08 6D 65 73 73 61 67 65 31 70 74 00 get..message1pt.
02B0 28 6A 61 76 61 78 2E 66 61 63 65 73 2E 63 6F 6D (javax.faces.com
02C0 70 6F 6E 65 6E 74 2E 68 74 6D 6C 2E 48 74 6D 6C ponent.html.Html

- Power analysis (SPA/DPA)
- Electromagnetic analysis
- Glitches (clock, power supply, data corruption)
- Laser cutting and fault injection
- Focused ion beam surgery, etc.






Leakage resilient?

Leakage-resilient cryptography

New research field developed by Pietrzak et al. (2008+)

Definition of schemes more resistant to side channels

Leakage modelized by a **leakage function**
that is independent of the type of attack

(a 2-minute tutorial: `http://www.youtube.com/watch?v=89K3j_Rsbco`)

Examples of models (leakage functions)

Exposure-resilience

- Aims to model cold boot attacks (say)
- Leakage $= F(\text{memory})$

Private circuits

- Aims to model probing attacks
- Leakage $=$ values of any t circuit wires

Examples of models (leakage functions)

Bounded leakage

- Aims to model leakage of computation
- Leakage $= F$(input, secret, randomness),
  $F : \{0, 1\}^\star \rightarrow \{0, 1\}^\lambda$

Bounded retrieval

- Aims to model malware attacks
- Complete control of software and hardware
- Limited bandwidth available

Should we care?

		
Leakage	*Leakage* Leakage	*"Leakage"* Leakage

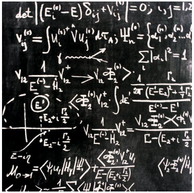
- Big gap between models and reality
- A leakage-resilient mode was broken. . . by DPA

OTOH:

- It may be the “best effort” on the algorithm side
- Co-design algorithm/implementation necessary

Part 2: Algorithmic attacks

- State-of-the-ciphers
- Why attacks aren't attacks
- Cognitive biases
- An attack that works
- What about AES?



ALGORITHMIC ATTACKS = attacks targetting a cryptographic function seen **as an algorithm** and **described as algorithms** rather than as physical procedures

Independent of the implementation!

Focus on **symmetric** cryptographic functions:

- Block ciphers
- Stream ciphers
- Hash functions
- PRNGs
- MACs

Low-impact attacks

Block ciphers:

- **AES**
- **GOST** (Russian standard, 1970's!)
- **IDEA** (1991)
- **KASUMI** (3GPP)

Hash functions:

- **SHA-1**
- **Whirlpool** (ISO)

Medium- to high-impact attacks

Block cipher:

- **DES** (56-bit key): practical break by… bruteforce

Stream cipher:

- **A5/1** (GSM): attacks on GSM, commercial "interceptors"

Hash function:

- **MD5**: rogue certificate attack PoC

Unattacked primitives

Block ciphers

- **CAST5** (default cipher in OpenPGP)
- **IDEA NXT** (a/k/a FOX)
- **Serpent, Twofish** (AES finalists)

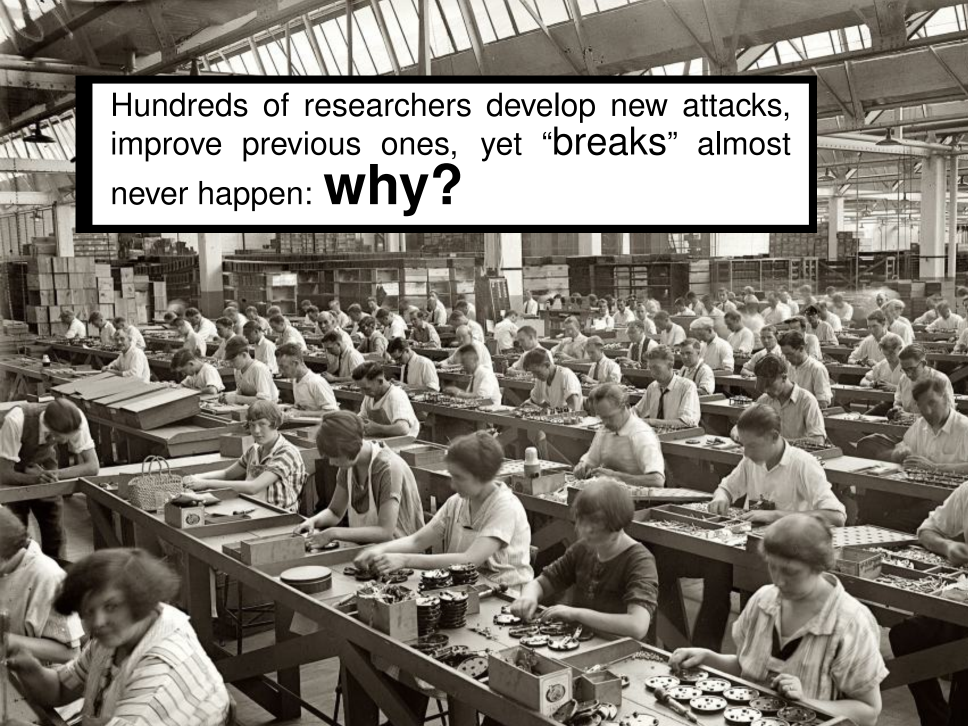
Stream ciphers:

- **Grain128a** (for hardware)
- **Salsa20** (for software)

Hash functions:

- **SHA-2** (SHA-256, . . . , SHA-512)
- **RIPEMD-160** (ISO std)

Hundreds of researchers develop new attacks, improve previous ones, yet "**breaks**" almost never happen: **why?**



#1: Insanely high time complexities

Example: preimage attack on **MD5** with time complexity

$$2^{123} \quad \text{(against } 2^{128} \text{ ideally)}$$

MD5 can no longer claim 128-bit security...

How (more) practical is a 2^{123} complexity?

Back-to-reality interlude

2 GHz CPU

$\Rightarrow$ 1 sec = $2 \cdot 10^9 \approx \mathbf{2^{33}}$ clocks

1 year	2^{58}	clocks
1000 years	2^{68}	clocks
since the Big-Bang	2^{116}	clocks

The difference between 80 bits and 128 bits of keysearch is **like the difference between a mission to Mars and a mission to Alpha Centauri**. As far as I can see, there is *no* meaningful difference between 192-bit and 256-bit keys in terms of practical brute force attacks; **impossible is impossible**.

John Kelsey, NIST hash-forum list

#2: Building blocks

Example: 2^{96} collision attack on the compression function of the SHA-3 candidate **LANE**

- Did not lead to an attack on the hash
- Invalidates a security proof (not the result!)
- Disqualified LANE from the SHA-3 competition

#2: Building blocks

Example: 2^{96} collision attack on the compression function of the SHA-3 candidate **LANE**

- Did not lead to an attack on the hash
- Invalidates a security proof (not the result!)
- Disqualified LANE from the SHA-3 competition

How to interpret such attacks?

1. We attacked something $\Rightarrow$ it must be weak!
2. We failed to attack the function $\Rightarrow$ it must be strong!

#3: Strong models, like "related-keys"

Attackers learn encryptions with a derived key

$$K' = f(K)$$

Actually an old trick: when Enigma operators set rotors incorrectly, they sent again with the correct key...

Modern version introduced by Knudsen/Biham in 1992

Practical on weak key-exchange protocols (EMV, 3GPP?) but **unrealistic in any decent protocols**

Related-key attack example

Key-recovery on **AES-256** with time complexity

$$2^{99}$$

against 2^{256} ideally

Needs 4 related subkeys!

The attacks are still mainly of theoretical interest and do not present a threat to practical applications using AES
the authors (Khovratovich / Biryukov)

Real-world model: pay-TV encryption

MPEG stream encrypted with CSA
= Common Scrambling Algorithm, **48-bit** key

Useful break of CSA needs

- Unknown-fixed-key attacks
- **Ciphertext-only**, partially-known plaintext (no TMTO)
- Key recovery **in** $<$**10 seconds** ("cryptoperiod")

#4: Memory matters

Back to our previous examples:

- **MD5**: time $2^{123.4}$ and 2^{50}B memory (1024 TiB)
- **LANE**: time 2^{96} and 2^{93}B memory (2^{53} TiB)
- **AES-256**: time 2^{119} and 2^{77}B memory (2^{37} TiB)

Memory is not free! ($$$, infrastructure, latency)

New attacks should be compared to generic attacks **with a similar budget**

See Bernstein's *Understanding bruteforce*
`http://cr.yp.to/papers.html#bruteforce`

#5: Banana attacks

#5: Distinguishing attacks

Used to be statistical biases, now:

- Known- or chosen-key attacks (!)
- Sets of input/output's satisfying some relation
- Anything "unexpected"

You-know-what-I-mean attacks (Daemen)

Example: **zero-sum attacks** on a block cipher E_K:

- Find inputs $X_1, X_2, \ldots, X_n$ such that

$$X_1 \oplus X_2 \oplus \cdots \oplus X_n = E_K(X_1) \oplus E_K(X_2) \oplus \cdots \oplus E_K(X_n) = 0$$

Attacks vs. Reality

2 interpretations of theoretical attacks:

1. Vulnerability that may be exploited
2. Evidence of no effective attack

Why can we be biased?

Cryptographic Numerology

The basic concept is that as long as your encryption keys are at least “this big”, you’re fine, even if none of the surrounding infrastructure benefits from that size or even works at all

Ian Grigg, Peter Gutmann, IEEE Security & Privacy 9(3), 2011

Cryptographic Numerology

The basic concept is that as long as your encryption keys are at least “this big”, you’re fine, even if none of the surrounding infrastructure benefits from that size or even works at all

Ian Grigg, Peter Gutmann, IEEE Security & Privacy 9(3), 2011

Choosing a key size is fantastically easy, whereas making the crypto work effectively is really hard

Ibid

Zero-risk bias

Preference for reducing a small risk to zero over a greater reduction in a larger risk

Example: reduce risk from 1% to 0% whereas another risk could be reduced from 50% to 30% at the same cost

Zero-risk bias

Preference for reducing a small risk to zero over a greater reduction in a larger risk

Example: reduce risk from 1% to 0% whereas another risk could be reduced from 50% to 30% at the same cost

Cryptographic numerology (examples)

- 1% = scary-new attack threat
- Move from 1024- to 2048-bit (or 4096-bit!) RSA
- Cascade-encryption with AES + Serpent + Twofish

+ **Unintended consequences**:
Crypto is slower $\Rightarrow$ less deployed $\Rightarrow$ less security

A *selection bias*: We will find the average height of Americans based on a sample of NBA players

Survivorship bias

= another *selection bias*

We **only see the unbroken ciphers**

We don’t see all the experimental designs broken in the course of the evaluation process

Survivorship bias

= another *selection bias*

We **only see the unbroken ciphers**

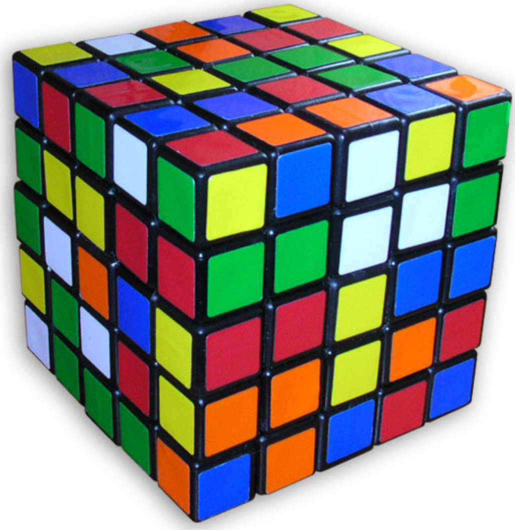
We don't see all the experimental designs broken in the course of the evaluation process

Example: **56 SHA-3 submissions** published

- **14 implemented** attacks (e.g. example of collision)
- **3 close-to-practical** attacks ($\approx 2^{60}$)
- **14 high-complexity** attacks

$\Rightarrow$ Attacks kill ciphers before they are deployed

An attack that works in reality



Cube attack

By Dinur and Shamir (2008)

- Refined high-order differential attack
- Black-box attack (fixed secret key)
- Precomputation + online stage

Complexity is **practical** and **experimentally verified**

The attack relies on empirical observations:

- Algrebraic degree of implicit equations
- Structure of derivative equations

Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128

Jean-Philippe Aumasson[1,*], Itai Dinur[2], Luca Henzen[3], Willi Meier[1,†], and Adi Shamir[2]

[1] FHNW, Windisch, Switzerland
[2] Weizmann Institute, Rehovot, Israel
[3] ETH Zurich, Switzerland

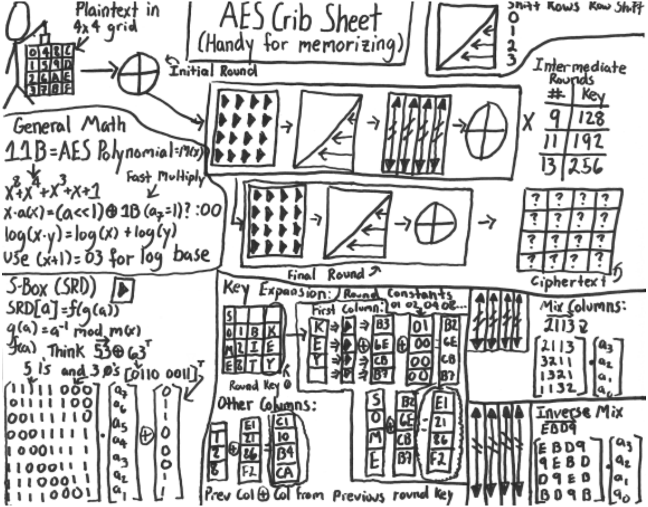
Abstract. Cube testers are a generic class of methods for building distinguishers, based on cube attacks and on algebraic property-testers. In this paper, we report on an efficient FPGA implementation of cube testers on the stream cipher Grain-128. Our best result (a distinguisher on Grain-128 reduced to 237 rounds, out of 256) was achieved after a computation involving 2^{54} clockings of Grain-128, with a 256×32 parallelization. An extrapolation of our results with standard methods suggests the possibility of a distinguishing attack on the full Grain-128 in time 2^{83}, which is well below the 2^{128} complexity of exhaustive search. We also describe the method used for finding good cubes (a simple evolutionary algorithm), and report preliminary results on Grain-v1 obtained with a bitsliced C implementation. For instance, running a 30-dimensional cube tester on Grain-128 takes 10 seconds with our FPGA machine, against about 45 minutes with our bitsliced C implementation, and more than a day with a straightforward C implementation.

An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware

Itai Dinur[1], Tim Güneysu[2], Christof Paar[2],
Adi Shamir[1], and Ralf Zimmermann[2]

[1] Computer Science department, The Weizmann Institute, Rehovot, Israel
Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany

Abstract. In this paper we describe the first single-key attack which can recover the full key of the full version of Grain-128 for arbitrary keys by an algorithm which is significantly faster than exhaustive search (by a factor of about 2^{38}). It is based on a new version of a cube tester, which uses an improved choice of dynamic variables to eliminate the previously made assumption that ten particular key bits are zero. In addition, the new attack is much faster than the previous weak-key attack, and has a simpler key recovery process. Since it is extremely difficult to mathematically analyze the expected behavior of such attacks, we implemented it on RIVYERA, which is a new massively parallel reconfigurable hardware, and tested its main components for dozens of random keys. These tests experimentally verified the correctness and expected complexity of the attack, by finding a very significant bias in our new cube tester for about 7.5% of the keys we tested. This is the first time that the main components of a complex analytical attack are successfully realized against a full-size cipher with a special-purpose machine. Moreover, it is also the first attack that truly exploits the configurable nature of an FPGA-based cryptanalytical hardware.


AES Crib Sheet
(Handy for memorizing)
Plaintext in 4x4 grid
Initial Round
Shift Rows Row Shift
Intermediate Rounds
Key
9 128
11 192
13 256
Final Round
Ciphertext
General Math
11B=AES Polynomial=m(x)
Fast Multiply
$x^8+x^4+x^3+x+1$
$x \cdot a(x) = (a<<1) \oplus 1B\ (a_7=1)?:00$
$\log(x \cdot y) = \log(x) + \log(y)$
Use (x+1) = 03 for log base
S-Box (SRD)
SRD[a] = f(g(a))
$g(a) = a^{-1} \bmod m(x)$
f(a) Think 53 ⊕ 63
5 1's and 3 0's [0110 0011]
Key Expansion: Round Constants 01 02 04 08...
First Column:
Round Key 0
Other Columns:
Prev Col ⊕ Col from Previous round key
Mix Columns: 2113
Inverse Mix EBD9

www.theregister.co.uk/2011/08/19/aes_crypto_attack/

Login | Sign up

The Register®

Hardware Software Music & Media Networks Security Cloud Public Sector Business Scie

Crime Malware Enterprise Security Spam ID Compliance

Print Tweet Alert

AES crypto broken by 'groundbreaking' attack

Faster than simply brute-forcing

By Dan Goodin in San Francisco • Get more from this author

Posted in Security, 19th August 2011 05:00 GMT

Updated Cryptographers have discovered a way to break the Advanced Encryption Standard used to protect everything from top-secret government documents to online banking transactions.

Groundbreaking attack!

How badly is AES broken?

The **facts**:

- AES-128: 2^{126} complexity, 2^{88} plaintext/ciphertext
  against 2^{128} *and* 2^{0} *for bruteforce*
- AES-256: 2^{254} complexity, 2^{40} plaintext/ciphertext
  against 2^{256} *and* 2^{1} *for bruteforce*

See Bogdanov, Khovratovich, Rechberger:

`http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf`

How badly is AES broken?

The **facts**:

- AES-128: 2^{126} complexity, 2^{88} plaintext/ciphertext
  against 2^{128} *and* 2^{0} *for bruteforce*
- AES-256: 2^{254} complexity, 2^{40} plaintext/ciphertext
  against 2^{256} *and* 2^{1} *for bruteforce*

See Bogdanov, Khovratovich, Rechberger:

`http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf`

Reactions heard (e.g. from customers):

- **AES is insecure**, let's do at least 50 rounds!
- **AES is always secure**, because it's the standard!

Conclusion

Real threats are physical/implementation/OPSEC attacks

- Bad implementation, misuse, side channels, passwords, etc.

Leakage-resilient crypto of little help so far

Real threats are physical/implementation/OPSEC attacks

- Bad implementation, misuse, side channels, passwords, etc.

Leakage-resilient crypto of little help so far

Algorithmic attacks break ciphers before we use them, thus are a not a significant threat

We don't break the codes, we try to analyze how secure they are

– Orr Dunkelman, panel on security, 2011

Real threats are physical/implementation/OPSEC attacks

- Bad implementation, misuse, side channels, passwords, etc.

Leakage-resilient crypto of little help so far

Algorithmic attacks break ciphers before we use them, thus are a not a significant threat

We don't break the codes, we try to analyze how secure they are

– *Orr Dunkelman, panel on security, 2011*

When deploying crypto, beware cognitive biases!

Real threats are physical/implementation/OPSEC attacks

- Bad implementation, misuse, side channels, passwords, etc.

Leakage-resilient crypto of little help so far

Algorithmic attacks break ciphers before we use them, thus are a not a significant threat

We don't break the codes, we try to analyze how secure they are

– Orr Dunkelman, panel on security, 2011

When deploying crypto, beware cognitive biases!

AES is fine, weak implementations are the biggest threat

The encryption doesn't even have to be very strong to be useful, it just must be **stronger than the other weak links** in the system. Using any standard commercial risk management model, cryptosystem failure is orders of magnitude below any other risk.

Ian Grigg, Peter Gutmann, IEEE Security & Privacy 9(3), 2011

If you think like an attacker, then you're a fool to worry about the crypto. Go buy a few zero days.

Jon Callas, randombit.net cryptography list, 2011

THANK YOU!

Please complete the feedback form ☺