Cryptanalysis vs. Reality

Jean-Philippe 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*

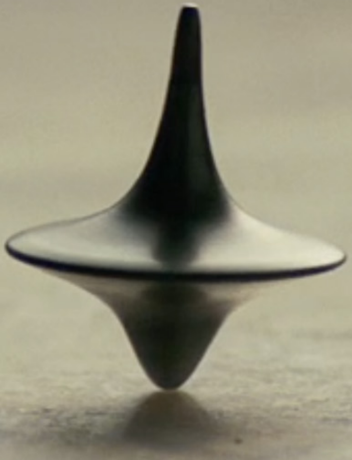
The fundamental goal of a cryptanalyst is to violate one or several security notions for algorithms that claim, implicitly or explicitly, to satisfy these security notions.

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.

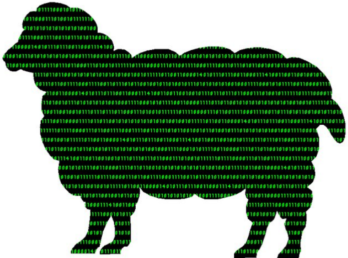
Compact Oxford English Dictionary



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

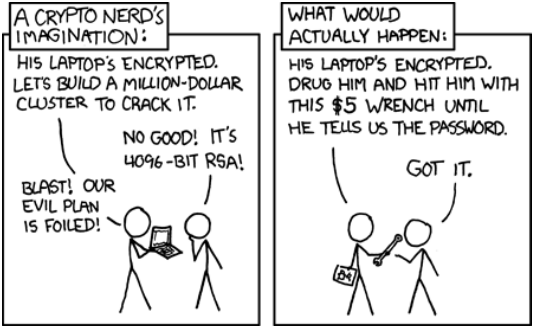
All models are in **simulacra**, that is, simplified reflections of reality, but, despite their inherent falsity, they are *nevertheless extremely useful*

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

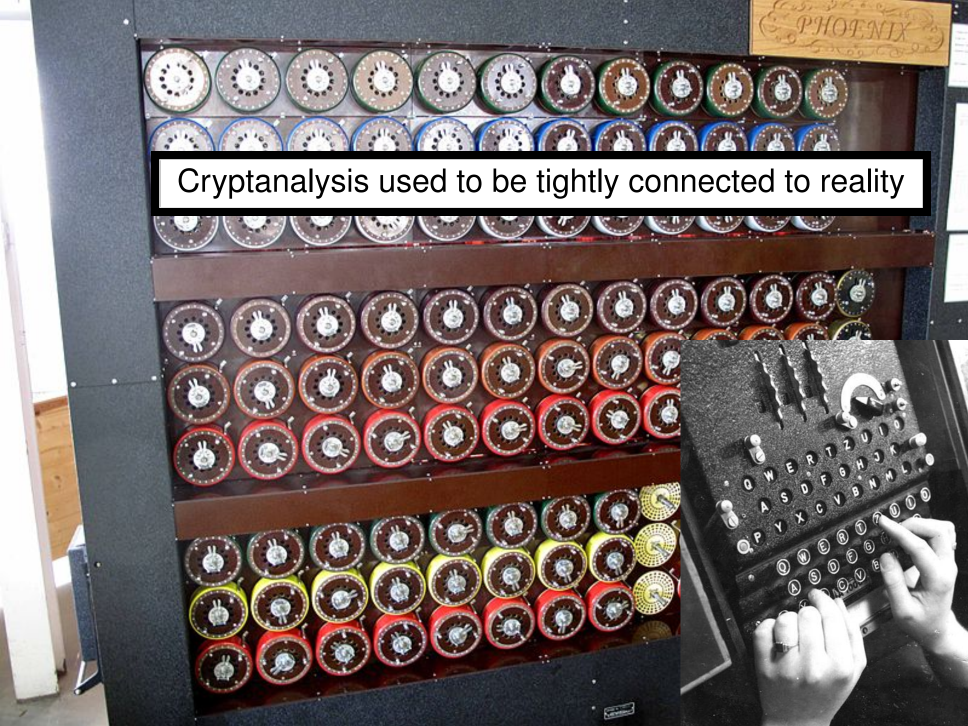


Cryptanalysis usually excludes methods of attack that do not primarily target weaknesses in the actual cryptography, such as bribery, physical coercion, burglary, keystroke logging, and social engineering, although these types of attack are an important concern and are often more effective

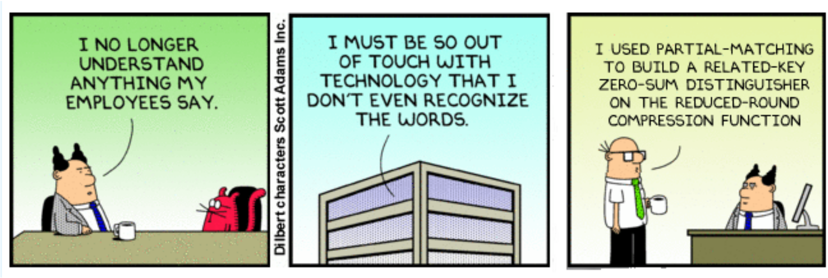
Wikipedia



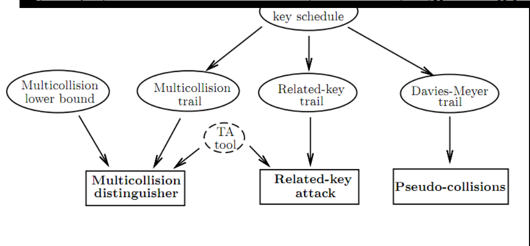
Cryptanalysis used to be tightly connected to reality



But 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

4 Comments

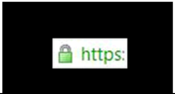
A group of cryptographers has developed a new attack that has broken Kasumi, the encryption algorithm used to secure traffic on 3G GSM wireless networks. The technique enables them to recover a full key by using a tactic known as a related-key attack, but experts say it is not the

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

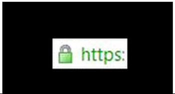
A group of cryptographers has developed a new attack that has broken Kasumi, the encryption algorithm used to secure traffic on 3G GSM wireless networks. The technique enables them to recover a full key by using a tactic known as a related-key attack, but experts say it is not the

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).

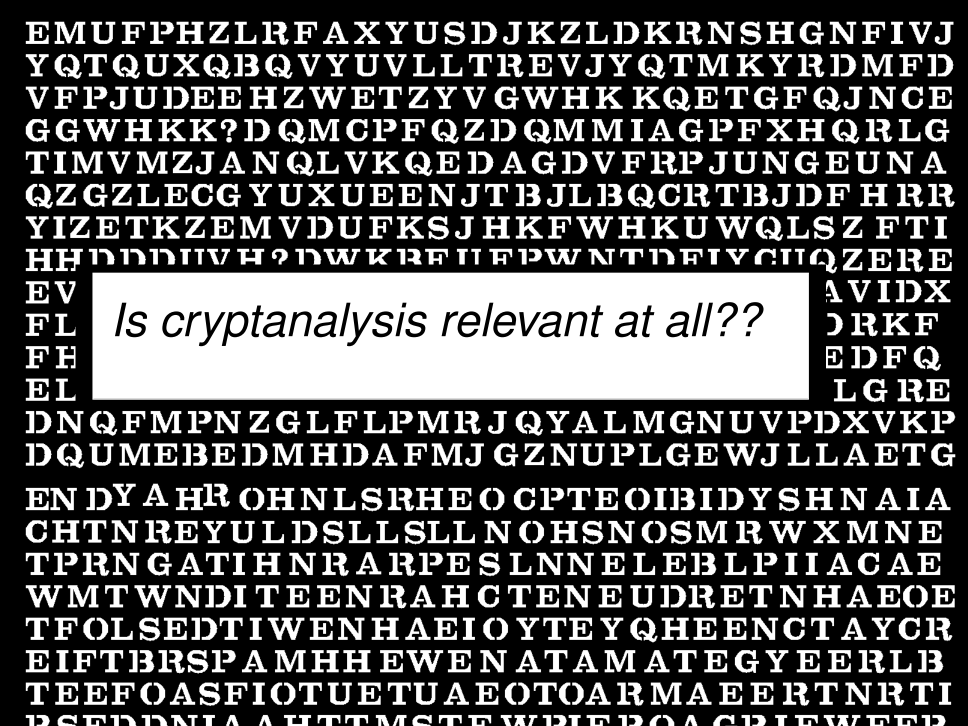


Models' language overlaps with real-world language: "attacks", "broken" have multiple meanings

Has cryptanalysis 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??

Remainder of this talk

PART 1: PHYSICAL ATTACKS

- Bypass and misuse
- Side channels

PART 2: ALGORITHMIC ATTACKS

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

CONCLUSIONS + REFERENCES

PART 1: PHYSICAL ATTACKS

- Bypass and misuse
- Side channels

HTTPS protection uses (say) **2048-bit RSA** to authenticate servers, and to avoid MitM attacks

$\approx$ 100-bit security (see `http://www.keylength.com/`)

$\Rightarrow \approx 2^{100}$ ops to break RSA by factoring the modulus

HTTPS protection uses (say) **2048-bit RSA** to authenticate servers, and to avoid MitM attacks

$\approx$ 100-bit security (see `http://www.keylength.com/`)

$\Rightarrow \approx 2^{100}$ ops to break RSA by factoring the modulus

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

HTTPS protection uses (say) **2048-bit RSA** to authenticate servers, and to avoid MitM attacks

$\approx$ 100-bit security (see `http://www.keylength.com/`)

$\Rightarrow \approx 2^{100}$ ops to break RSA by factoring the modulus

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

Or 2^0 by compromising a 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
```

AES-256 provides 256-bit security (does it really?)

FIPS 140-2 is supposed to inspire confidence…

Yet “secure” USB drives by Kingston, SanDisk, Verbatim were easily broken

The flaw: password validation on host PC
+ static unlock code

How NOT to use decent cryptographic primitives:

How NOT to use decent cryptographic primitives:

ECDSA signing with a constant instead of a random number

to find SONY PS3's private key

How NOT to use decent cryptographic primitives:

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 (as found in the WEP WiFi "protection")

How NOT to use decent cryptographic primitives:

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 (as found in the WEP WiFi "protection")

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

Software side-channel attacks

Practical attacks exploiting non-constant-time AES implementations

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 Go

FORMS has 1 elements

Step 2

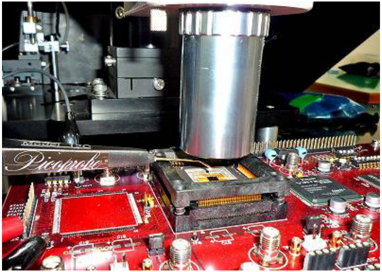
Form	Field	Type	Value
form	form:input1	text	
form	form:button1	submit	press me
form	autoScroll	hidden	
form	form_SUBMIT	hidden	1
form	form:_link_hidden_	hidden	
form	form:_idcl	hidden	
form	javax.faces.ViewState	hidden	9JgUKANlia8gDSeJJ6dfgYtl3C3vAXPnXVlCITj3uBAIyrV5uUsjPylY1EfrDAiDZOFVD/ZKqh3XlxjJD3JfR0g0Kr

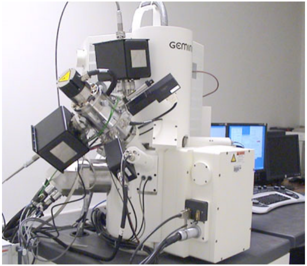
Stop Decrypting Decryption finished!

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

Hardware side-channel attacks

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





PART 2: ALGORITHMIC ATTACKS

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

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

ALGORITHMIC ATTACKS are thus **independent of the implementation** of the function attacked

We'll focus on **symmetric** cryptographic primitives:

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

Though there'd be a lot to say about public-key encryption/signatures, authentication protocols, etc.

Null- to low-impact attacks (examples)

Block ciphers:

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

Hash functions:

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

Medium- to high-impact attacks (examples)

Block cipher:

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

Stream cipher:

- **A5/1** (GSM): attacks on GSM facilitated

Hash function:

- **MD5**: famous rogue certificate attack PoC

Unattacked primitives (examples)

Block ciphers

- **CAST5** (default cipher in OpenPGP)
- **IDEA** (1991!)
- **IDEA-NXT** (aka FOX)
- **Serpent** (AES finalist)
- **Twofish** (AES finalist)

Stream ciphers:

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

Hash functions:

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

Despite the large amount of research and new techniques, “breaks” almost never happen: **Why?**

High-complexity attacks

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

$$2^{123.4}$$

against 2^{128} ideally

High-complexity attacks do not matter as long as

- the effort is obviously unfeasible, or
- overwhelms the cost of other attacks

Yet MD5 can no longer be sold as "128-bit security" hash

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)

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

Attacks on 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 the security reduction compression$\prec$hash
- Disqualified LANE from the SHA-3 competition!

Attacks on 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 the security reduction compression$\prec$hash
- Disqualified LANE from the SHA-3 competition!

How to interprete those attacks?

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

Strong models: ex of **related-key** attacks

Attackers learn encryptions with a derived key

$$K' = f(K)$$

One of the first attacks: 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 most decent protocols**

Related-key attacks example

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

$$2^{119}$$

against 2^{256} ideally!

Needs 4 related keys… actually, related subkeys!

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

Model from reality: pay-TV encryption

MPEG stream encrypted with CSA
Common Scrambling Algorithm, 48b or 64b key

Useful break of CSA needs

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

There's not only time!

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)

Practical cost of access to memory neglected

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

See "cracking machines" in *Understanding bruteforce*
`http://cr.yp.to/papers.html#bruteforce`

Distinguishing attacks

aka **distinguishers**

Used to be statistical biases

Now distinguishers are

- Known- or chosen-key attacks
- Sets of input/output's satisfying some relation

Example: differential q-multicollision distinguisher on **AES**

$$\begin{aligned} E_{K_1}(P_1) \oplus E_{K_1 \oplus \Delta}(P_1 \oplus \nabla) &= E_{K_2}(P_2) \oplus E_{K_2 \oplus \Delta}(P_2 \oplus \nabla) \\ &= E_{K_3}(P_3) \oplus E_{K_3 \oplus \Delta}(P_3 \oplus \nabla) = \dots \end{aligned}$$

Distinguishing attacks

aka **distinguishers**

Used to be statistical biases

Now distinguishers are

- Known- or chosen-key attacks
- Sets of input/output's satisfying some relation

Example: differential q-multicollision distinguisher on **AES**

$$\begin{aligned} E_{K_1}(P_1) \oplus E_{K_1 \oplus \Delta}(P_1 \oplus \nabla) &= E_{K_2}(P_2) \oplus E_{K_2 \oplus \Delta}(P_2 \oplus \nabla) \\ &= E_{K_3}(P_3) \oplus E_{K_3 \oplus \Delta}(P_3 \oplus \nabla) = \dots \end{aligned}$$

NO IMPACT ON SECURITY in a large majority of cases

Attacks (high-complexity, strong model, high-memory, distinguishers, etc.) **vs. Reality**

2 general interpretations:

1. This little thing is a sign of bigger things!
2. This little thing is a sign of no big things!

Why are we biased? (towards 1. or 2.)

Cryptographic Num3rol0gy

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

Cryptographic Num3rol0gy

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

Choosing a key size if 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 ⇒ less deployed ⇒ less security

Survivorship bias

We **only remember/see the unbroken**, deployed and/or standardized, algorithms

Not the numerous experimental designs broken

Survivorship bias

We **only remember/see the unbroken**, deployed and/or standardized, algorithms

Not the numerous experimental designs broken

Example: of the **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

$\Rightarrow$ Practical attacks kill ciphers before they are used and known to the public

What about AES?

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

Free whitepaper – IBM System Networking RackSwitch G8124

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.

What about AES?

Groundbreaking attack bogeyman!

What about AES?

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`

What about AES?

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 (from customers, third parties):

- **AES is insecure!** Let's use AES with 42 rounds!
- **AES is secure!** The attack is far from practical!

CONCLUSIONS + REFERENCES

Conclusions

Algorithmic attacks on deployed schemes are (almost) never a threat to security, due to

- High complexities, unrealistic models, etc.
- Weak ciphers are broken earlier and forgotten

We don't break ciphers, we evaluate their security

Orr Dunkelman

Conclusions

Algorithmic attacks on deployed schemes are (almost) never a threat to security, due to

- High complexities, unrealistic models, etc.
- Weak ciphers are broken earlier and forgotten

We don't break ciphers, we evaluate their security

Orr Dunkelman

Beware cryptographic numerology!

Conclusions

Algorithmic attacks on deployed schemes are (almost) never a threat to security, due to

- High complexities, unrealistic models, etc.
- Weak ciphers are broken earlier and forgotten

We don’t break ciphers, we evaluate their security

Orr Dunkelman

Beware cryptographic numerology!

AES is fine, weak implementations are the biggest threat

Related works

Leakage-resilience vs. Reality

Leakage Resilient Cryptography in Practice

Standaert et al. `http://eprint.iacr.org/2009/341`

Bruteforce vs. Reality

Using the Cloud to Determine Key Strengths

Kleinjung et al. `http://eprint.iacr.org/2011/254`

Crypto libs vs. Reality

Open-Source Cryptographic Libraries and Embedded Platforms

Junod `http://crypto.junod.info/hashdays10_talk.pdf`

Thank you for your attention