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 [Jou09, p3]

The goal of a cryptanalyst can be to

- Find the secret key of a block cipher
- Find a collision for a hash function
- Forge a valid digital signature or MAC
- Find a statistical bias in the output of a stream cipher
- Impersonate a party in an authentication protocol
- Influence the outcome of a key-agreement protocol
- Learn the inputs of third-parties’ in a multi-party computation
- Extract information from a zero-knowledge proof

Cryptanalysis is thus much more than mere codebreaking. Whereas in ancient times cryptography schemes only aimed to make messages unintelligible, cryptographers deal today with sophisticated schemes, and thus with complex security notions.

Reality is complex too. In our everyday lives we all resort to dramatic simplifications of reality to make decisions, be it when investing money, deciding on a new job, etc.. So do cryptanalysts, who have to consider simplified attackers.

Such a simplification is done through an attack model, which establishes the assumptions on what the attacker can and cannot do. Attack models are just a formal and rigorous way of simplifying reality. They usually exclude methods that target the user or its environment rather than the crypto schemes, hence physical coercion, social engineering, keylogging, or bribery are considered as out of scope.

As security notions and attacks are increasingly sophisticated, so do the attack models. State-of-the-art research often defines as “attacks” results that are only remotely related to a meaningful notion of security. This terminology gap has sometimes led to confusions, for example when media report headlines such as “AES encryption is cracked”\(^1\).

\(^1\)http://www.theinquirer.net/inquirer/news/2102435/aes-encryption-cracked
One may thus wonder whether cryptanalysis has lost connection with reality, and even whether it is relevant at all, for there are generally much simpler ways of penetrating the security system. This white paper (and its companion talk) is an attempt to address this issue by 1) providing a “reality-check” to those overestimating the role of cryptanalysis, and 2) showing that cryptanalysis does play an important role in the security process.

### Bypass and misuse of crypto

*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.*


The strength of crypto algorithms generally overwhelms that of other components of a security system. This fact is best illustrated with examples:

- X.509 digital certificates rely on the strength of crypto algorithms such as 2048-bit RSA or SHA-1, yet their security has been compromised by exploiting flawed certification path validation [Mar09], or directly by breaching the network of a certificate authority to forge rogue certificates [Bri11].
- Secure USB drives that stored data encrypted with AES-256 were attacked due to an insecure software-based password validation [KH10], allowing one to access decrypted data without knowing the key or the password.

In these two examples, the crypto was not broken, but rather bypassed. In other words, the attacks were independent of the ciphers chosen.

Sometimes a security system fails when strong crypto is misused, rather than bypassed, like in these two major failures:

- The private key in Sony’s PlayStation 3, as used to produce ECDSA signatures of legitimate code, could be recovered due to the reuse of a same number in place of a random number [fai10].
- The WEP WiFi protection is insecure due to its use of the stream cipher RC4 with a secret key that includes not only secret data, but also public and predictable data [FMS01].

Nevertheless, ECDSA and RC4 are strong algorithms, and not known to be broken when properly used.

### Side-channel attacks

Side-channel attacks target implementations of cryptographic algorithms, rather than only the mathematical object. They exploit the leakage of data from physical processes of the implementation, be it through measurement execution time, power consumption, electromagnetic emanation,
or through behavior under critical environment, etc. It is known that virtually any algorithm, regardless of its strength against algorithmic attack, can fail due to side-channel attacks on unprotected implementations.

Even pure software side-channel attacks can be very powerful. For example, the implementation of AES in OpenSSL 0.9.8n, then allegedly protected against cache-timing attacks [Ber05a, TOS10], was eventually compromised [BGK10].

Hardware-based attacks, however, are much more powerful than pure software side-channel attacks. Capabilities of invasive attacks are almost unlimited, provided one has a sufficient budget: electron microscopes allow submicron imaging of hardware circuits; laser technologies can be used to inject faults and to assist microprobing; powerful focused ion beam workstations perform chip-level “surgery” with extremely high precision.

As a response to certain classes of side-channel attacks, cryptography theorists introduced the notion of leakage-resilient cryptography [DP08, FKPR10, DP10]. New schemes were presented, such as leakage-resilient signature schemes, with a rigorous proof that the system remains secure even in the presence of an adversary exploiting physical leakage. It is unclear, however, whether the attack models considered capture real-world attackers [SPY 09].

State-of-the-art attacks

The recent years have been rich in new cryptanalysis techniques, in part thanks to the organization of two “cryptographic demolition derbies”:

- **eSTREAM, the ECRYPT Stream Cipher Project** was ran by the EU-funded Network of Excellent ECRYPT from 2004 to 2008. eSTREAM gathered partners from industry and academia, and succeeded in identifying promising new stream ciphers like Grain [HJMM08] or Salsa20 [Ber08]. eSTREAM received 35 candidate submissions, and more than 200 research reports were published about the design, analysis, or implementation of the candidates.

- **The SHA-3 Hash Competition**, organized by the US National Institute of Standards and Technology (NIST) started in 2008 and will select in 2012 the new hash function standard SHA-3. NIST received 63 submissions from all around the world, by cryptographers from academia, industry, and government organizations. As of Dec 2011, the five finalist algorithms are BLAKE, Grøstl, JH, Keccak, and Skein.

Two of the most interesting cryptanalysis techniques of the recent years were developed in the context of the above competitions: cube attacks for key-recovery attacks on stream ciphers, and rebound attacks for collisions on hash functions. Below we informally present those new attacks, highlighting their unique characteristics and sketching their principles.

Cube attacks

Introduced by Dinur and Shamir in 2008 [DS09a], cube attacks are key-recovery attacks that only need black-box queries to the algorithm attacked. They build on previous higher-order differential

---

2 [http://www.ecrypt.eu.org/stream](http://www.ecrypt.eu.org/stream)

3 [http://www.nist.gov/hash-competition](http://www.nist.gov/hash-competition). See also the “SHA-3 zoo” [http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo](http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo)
cryptanalysis techniques. Variants and refinements of cube attacks include cube testers [ADMS09], side-channel cube attacks [DS09b], or dynamic cube attacks [DS11]. High-complexity cube attacks were successfully implemented in FPGA devices [ADH+, DGP+11].

Cube attacks rely on the following observations:

1. Any cipher with \( k \)-bit key \( K \) and \( n \)-bit plaintext \( P \) can be described as a sequence of binary equations. A very simplified example with \( k = n = 3 \) is

   \[
   \begin{align*}
   C_1 &= K_1 + P_2 + K_1P_2 + K_2K_3 + P_2P_3 \\
   C_2 &= 1 + K_2 + K_3 + P_1 + K_2P_1P_2 \\
   C_3 &= K_3 + P_1 + P_2 + K_1P_2 + K_2P_2
   \end{align*}
   \]

   where \( C_1, C_2, C_3 \) are the ciphertext bits, and where the sum is an XOR. In our example, equations are sparse (few monomials) and of low-degree. For real algorithms, the equations are so big that computing such algebraic expression is all but practically impossible.

2. One may reduce them to smaller equations by computing their derivatives. In our simple example, the derivative of \( C_3 \)'s equation with respect to \( P_2 \) is \( K_1 + K_2 \). Even for more complex algorithms, one may use derivatives of high-order to compute a value (one or zero) that is the XOR of a number of bits of key. For example, the derivative of \( C_2 \)'s equation with respect to \( P_1 \) and \( P_2 \) is

   \[
   \begin{align*}
   C_2(P_1 = 0, P_2 = 0) + C_2(P_1 = 1, P_2 = 0) + C_2(P_1 = 0, P_2 = 1) + C_2(P_1 = 1, P_2 = 1) \\
   &= (1 + K_2 + K_3) + (1 + K_2 + K_3 + 1) + (1 + K_2 + K_3) + (1 + K_2 + 1 + K_3 + K_2) \\
   &= K_2
   \end{align*}
   \]

   That is, the derivative with respect to the monomial in \( P \) of highest degree is its coefficient in \( K \) (here \( K_2 \)).

3. Although the equations obtained, such as \( K_1 + K_2 \), cannot be determined in advance—for the algebraic form of the implicit equations is unknown—one can use probabilistic linearity testing algorithms to determine whether the value obtained is a sum of key bits (rather than a combination of higher degree), and to identify the bits involved.

4. Once one has gathered a sufficient number of linear equations, the unknown key bits can be computed by solving the system using standard efficient methods (as Gauss' method).

A unique property of cube attacks, compared to most cryptanalytic attacks, is that their complexity is determined empirically, through experimental verification.

**Rebounds attacks**

Rebound attacks were introduced in 2009 [MRST09] as a general method to find collisions in hash functions. Contrary to cube attacks, rebound attacks need know the internals of the function attacked, which uses no secret key.

Rebound attacks combine advanced differential cryptanalysis techniques with meet-in-the-middle strategy. They optimize the exploit of freedom degrees in the search for inputs conforming to a differential characteristic, by attacking a permutation \( P(x) = P_1(P_2(P_3(x))) \) in two phases:

- An inbound phase, which searches for many solutions of the “hard differential” in \( P_2 \) using a meet-in-the-middle techniques.
An outbound phase, that tests whether one (or more) solution of the inbound phase satisfies the differential characteristic in $P_1$ and $P_3$.

Rebound attacks were applied to several SHA-3 candidates (e.g. [NP11, DGPW]), and generally have a high complexity. Unlike cube attacks, the cost of rebound attacks can be estimated analytically, and does not require experimental verification.

**Status of common algorithms**

At the time of writing, a number of widely deployed ciphers have been claimed to be broken yet their real-world security is all but unaltered. These include AES, the Russian standard GOST, KASUMI (as used in 3GPP), SHA-1, or the ISO standard hash Whirlpool. That is, the insecurity of those algorithms is so far only theoretical.

Other well-known algorithms, however, have seen their security degraded by advances in cryptanalysis: A5/1 (as used in GSM), or MD5. Nonetheless, many algorithms have so far resisted all kind of cryptanalysis. These are for example the block ciphers CAST5 (default in OpenPGP), IDEA, IDEA-NXT, the AES finalists Serpent and Twofish; the stream ciphers Grain-v1, Salsa20, SNOW 3G; the hash functions SHA-256, SHA-512 or RIPEMD-160.

**Five reasons why attacks aren’t attacks**

Algorithmic attacks are algorithms targetting a crypto function seen as an algorithm, rather than as a physical procedure. Algorithmic attacks are independent of the implementation of the function. Below we briefly describe some reasons behind the gap between attacks within cryptanalysts’ models, and real-world attacks.

**High complexities**

Security levels are expressed in terms of equivalent-key-bits: “$k$-bit security” means that the best attack requires the equivalent of $2^k$ basic operations. Therefore any attack method making fewer than $2^k$ operations is a theoretical break, even when the effort remains impractically high.

**Example:** A preimage attack with complexity $2^{123.4}$ was claimed [SA09] on the hash function MD5, whereas the generic attack has complexity $2^{128}$. To give an order of magnitude, note that a 2 GHz processor makes $\approx 2^{33}$ clock cycles per second, thus $\approx 2^{68}$ in 1000 years.

**Building blocks**

A number of attacks target a building block of a crypto function rather than the complete function. Although attacks on building blocks sometimes directly transfer to an attack on the function, they often don’t. Building blocks are for example the compression function of a hash function, or a reduced number of rounds of a block cipher.
Example: The SHA-3 candidate hash function LANE was eliminated from the competition after the publication of a high-complexity \(2^{96}\) instead of \(2^{128}\) collision attack on its compression function. This finding invalidated an argument used to “prove” the collision resistance of LANE, yet the \(2^{96}\) attack could not be used to attack LANE.

**Strong models**

Attack models sometimes give insanely high power to an attacker. Even the most standard model, which assumes that attackers can perform chosen-plaintext and chosen-ciphertext queries, is unrealistic in most real-life use cases—which is fortunate, for chosen-ciphertext queries are generally all an attacker needs. There are even stronger models:

- In the **related-key model**, the attacker is allowed to make chosen-plaintext and chosen-ciphertext queries for a modified key (like a modification of some bits).
- In the **known-key model**, the attacker knows the key, and aims to find non-randomness in the processing of a message.

Example: A key-recovery attack on AES-256 in \(2^{99}\) (instead of \(2^{256}\)) [BK09] required four related-keys, where the relation between the keys was expressed in terms of relation between round keys.

**Memory is not free**

Algorithmic attacks often need to store some precomputed data in large tables. Theoretical algorithm analysis clearly distinguishes time complexity from space complexity and assumes that fetching an element from memory takes negligible time. But in practice both memory and computations require physical hardware that has a price, and memory access times can be orders of magnitude higher than the time of a simple

Example: The aforementionned preimage attack on MD5 with \(2^{123.4}\) complexity requires a memory of \(2^{50}\) bytes (1024 tebibytes). Although relatively affordable, the use of such a large memory is likely to make any implementation of the attack considerably slower than a parallel brute-force search [Ber05b]. Even worse, the attack on LANE cited above requires memory for \(2^{93}\) bytes (\(2^{53}\) tebibytes!); although much smaller than the entropy of a supermassive black hole (\(\approx 2^{206}\) [FHRK09]), it means that the attack is impossible to implement without advanced alien technology.

**Distinguishers**

Attacks sometimes aim to detect “non-randomness” in a cipher or hash function, so as to distinguish it from an ideal function. Whereas previous so-called distinguishers consisted in finding a statistical bias in the output, recent works use more sophisticated notions of non-randomness that are often far from any notion of security.
Example:  A distinguisher [BK09] on AES, called $q$-multicollision, consisted in the finding of keys $K_1, \ldots, K_q$ and plaintexts $P_1, \ldots, P_q$ satisfying the following relation:

\[
E_{K_1}(P_1) \oplus E_{K_1\oplus\Delta}(P_1 \oplus \nabla) = E_{K_2}(P_2) \oplus E_{K_2\oplus\Delta}(P_2 \oplus \nabla) = E_{K_3}(P_3) \oplus E_{K_3\oplus\Delta}(P_3 \oplus \nabla) = \ldots
\]

As observed by the authors, such attacks “attacks are still mainly of theoretical interest and do not present a threat to practical applications using AES” [BK09].

Cognitive biases

There are two extreme interpretations of “attacks that are not attacks”:

1. the pessimistic one: the attack is the sign of more critical, yet undiscovered, vulnerabilities. As Bruce Schneier says, attacks always get better, never worse.

2. the optimistic one: such cryptanalytic attacks never break anything, so let’s just ignore them.

Below we attempt to identify some of the origins of such unreasonable attitudes, based on well-known cognitive biases.

Zero-risk bias

Zero-risk bias is the preference for reducing an already small risk to zero, over a greater reduction in a larger risk; the complete suppression of the risk is falsely perceived as of greater benefit.

Grigg and Gutmann recently discussed [GG11] a particular type of zero-risk bias that cryptographers often fall victim of, which they call cryptographic numerology and define as follows: “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.” They further argue that “choosing a key size if fantastically easy, whereas making the crypto work effectively is really hard.”

Due to this zero-risk bias, one may overestimate the risk induced by theoretical attacks. This may have unintended consequences as the overlook of bigger threats, or a degradation of performance caused by the use of “stronger” crypto.

Survivorship bias

Survivorship bias is the tendency to include only successes in statistical analysis, which often significantly skews the results. Applied to crypto, it occurs when one judges the relevance of cryptanalysis based on the attacks on algorithms that survived...cryptanalysis! Attacks on widely deployed schemes remain theoretical precisely because real-worldly-weak ones were broken in the course of the selection process.

A perfect example is the SHA-3 competition: of the 56 submissions published, efficient and practical attacks were found for 14 algorithms, and theoretical attacks were found for 14 other algorithms; in addition 3 close-to-practical ($\approx 2^{60}$) attacks were found. This illustrates the fact that practical attacks kill ciphers before they used and known to the public.
Conclusion

We have argued that, due to an increased sophistication of algorithmic attacks, there tends to be an embarrassing gap between theoretical and actual security of cryptographic algorithms. In practice, cryptography is often bypassed, and failures are caused by misuses, misimplementations, or implementation attacks, rather than algorithmic attacks. We should avoid cryptographic numerology and consider crypto as a component of the system rather than as the only attack surface.

Nonetheless, the importance of cryptanalysis should not be underestimated. Cryptanalysis succeeds in identifying weak ciphers before they are used, and it’s the reason why deployed schemes are seldom broken. After their deployment, cryptanalysis is necessary to evaluate the security of crypto algorithms, even if attacks remain theoretical. For example, the finding that a 256-bit key cipher only provides 128-bit security leads to performance gain, since switching to a (faster) 128-bit key cipher provides the same level of security.

References


