Snake Oil Crypto:

How I stopped to worry and started to love crypto

Team CIRCL
https://www.d4-project.org/

2019/11/27

Jean-Louis Huynen

Outline

- Cryptography 101,
- Encryption an Law Enforcement,
- Use-Case: RSA,
- First Hands-on: Understanding RSA,
- Snake-Oil-Crypto: a primer,
- Second Hands-on: RSA in Snake-Oil-Crypto,
- D4 passiveSSL Collection,
- Interactions with MISP.

Cryptography 101

- **Plaintext** P: Text in clear,
- **Encryption** E: Process of disguising the plaintext to hide its content,
- **Ciphertext** C: Result of the Encryption process,
- **Decryption** D: Process of reverting encryption, transforming C into P,
- **Encryption Key** EK: Key to encrypt P into C,
- **Decryption Key** DK: Key to decrypt C into P,
- **Cryptanalysis**: Analysis of C to recover P without knowing K.

- **Confidentiality** : Ensure the secrecy of the message except for the **intended** recipient,
- **Authentication** : Proving a party's identity,
- **Integrity** : Verifying that data transmitted were not altered,
- **Non-repudiation** : Proving that the sender sent a given message.

- **In-transit encryption**: protects data while it is transferred from one machine to another,
- **At-rest encryption**: protects data stored on one machine.

It [cipher] should not require secrecy, and it should not be a problem if it falls into enemy hands.

There is no security in obscurity.

Attackers model I

Black Box - Attackers may only see inputs / outputs:

- **Ciphertext-Only Attackers (COA) :** see only the ciphertext,
- **Known-Plaintext Attackers (KPA):** see ciphertext and plaintext,
- **Chosen-Plaintext Attacker (CPA):** encrypt plaintext, and see ciphertext,
- **Chosen-Ciphertext Attakers (CCA):** encrypt plaintext, decrypt ciphertext.

Grey Box - Attackers see cipher's implementation:

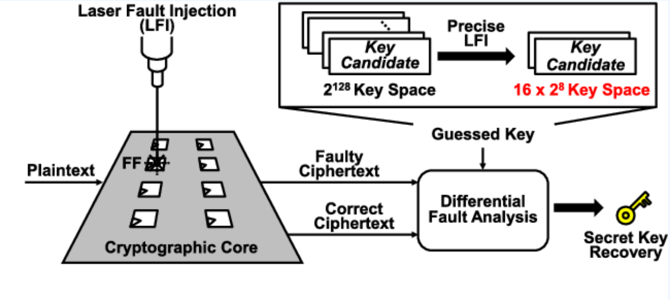
- **Side-Channel Attacks:** study the behavior of the implementation, eg. **timing attacks** [1]:
  - Osvik, Shamir, Tromer [OST06]: Recover AES-256 secret key of Linux's dmcrypt in just 65 ms
  - AlFardan, Paterson [AFP13]: "Lucky13" recovers plaintext of CBC-mode encryption in pretty much all TLS implementations
  - Yarom, Falkner [YF14]: Attack against RSA-2048 in GnuPG 1.4.13: "On average, the attack is able to recover 96.7% of the bits of the secret key by observing a single signature or decryption round."
  - Benger, van de Pol, Smart, Yarom [BvdPSY14]: "reasonable level of success in recovering the secret key" for OpenSSL ECDSA using secp256k1 "with as little as 200 signatures"

Attackers model III

Most recent timing attack: **TPM-fail** [24420]

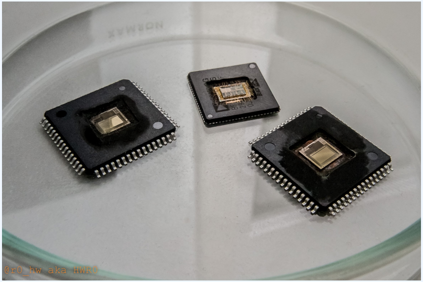
> We discovered timing leakage on Intel firmware-based TPM (fTPM) as well as in STMicroelectronics' TPM chip. Both exhibit secret-dependent execution times during cryptographic signature generation. While the key should remain safely inside the TPM hardware, we show how this information allows an attacker to recover 256-bit private keys from digital signature schemes based on elliptic curves.

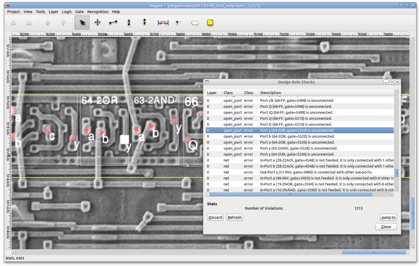
- **Invasive Attacks:**
  - injecting faults [MFS+18],



ATTACKERS MODEL V

- decapping chips [2], reverse engineering [3] [4], etc.





[1] https://cryptojedi.org/peter/data/croatia-20160610.pdf
[2] https://siliconpr0n.org/wiki/doku.php?id=decap:start
[3] http://siliconzoo.org
[4] http://degate.org

- **Indistinguishability (IND) :** Ciphertexts should be indistinguishable from random strings,
- **Non-Malleability (MD):** "Given a ciphertext $C_1 = E(K, P1)$, it should be impossible to create another ciphertext, C_2 , whose corresponding plaintext, P_2 , is related to P_1 in a meaningful way."

Semantic Security (IND-CPA) is the most important security feature:

- Ciphertexts should be different when encryption is performed twice on the same plaintext,
- To achieve this, randomness is introduced into encryption / decryption:
  - $C = E(P, K, R)$
  - $P = D(C, K, R)$

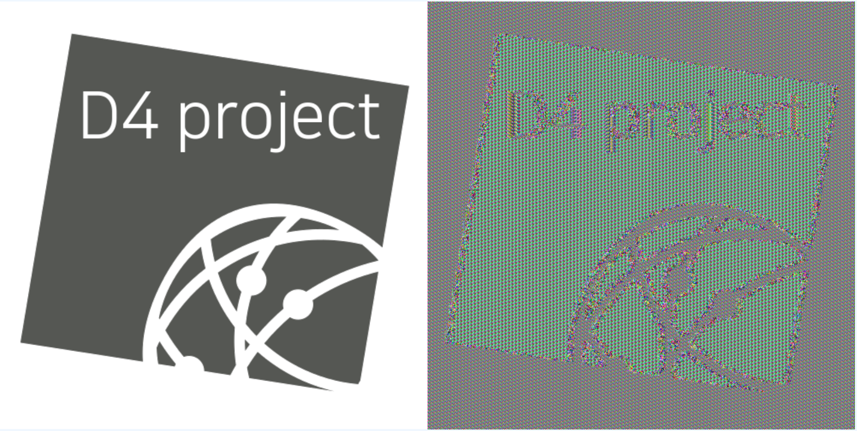


Figure: Image encrypted with AES-ECB

Semantic Security

IND-CPA should not leak information about the PlainText as long as the key is secret:

- $C^1 = E(K, P^1)$, $C^2 = E(K, P^2)$, what are the couples?
- the same message encrypted twice should return two different CipherText,
- one way to achieve this is to introduce randomness in the encryption process: $C = E(K, R, P)$ where R is fresh random bits,
- C should not be distinguishable from random bits.

No Semantic Security without randomness

Randomness

Random Number Generator:

-

Pseudo Random Number Generator:

-

Entropy

RSA 2048 is roughly 100 bits security.

-

Type of encryption

- Symmetric encryption,
- Asymmetric encryption.

Some attacks requires less than CCA / CPA:

- Side Channel attacks as for instance Padding Oracle (Vaudenay Attacks)

Encryption and Law Enforcement

- In the arms race between cryptographers and crypto-analysts. In terms of practical breaks, cryptographers are miles ahead.
- In a society that is ever more depending on the correct functioning of electronic communication services, technical protection of these service is mandatory,
- In the face of serious crimes, law enforcement may lawfully intrude privacy or break into security mechanisms of electronic communication,
- **proportionality** - collateral damages (class breaks)
- Resolving the encryption dilemma: collect and share best practices to circumvent encryption.

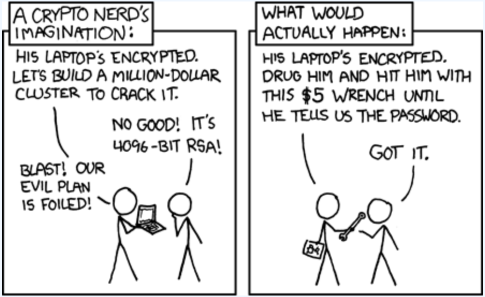
Any effort to reveal an unencrypted version of a target's data that has been concealed by encryption.

- **Try to get the key:**
  - **Find the key:**
    - physical searches for keys,
    - password managers,
    - web browser password database,
    - in-memory copy of the key in computer's HDD / RAM.
    - seize the key (keylogger).
  - **Guess the key:**,
    - Whereas encryption keys are usually too hard to guess (eg. 128bits security is 2^{128} trials (universe is 2^{88} ns old)),
    - passphrases are usually shorter to be memorizable, and are linked to the key,
    - some systems have limitations on sorts of passwords (eg. 4/6 digits banking application),
    - educated guess on the password from context,

Encryption Workarounds [KS17] II

- educated guess from owner's other passwords,
- dictionaries and password generation rules ([5]).
- Offline / online attacks (eg. 13 digits pw: 25.000 on an iphone VS matter of minutes offline),
- + beware devices protection when online (eg. iphone erase on repeated failures).

- **Compel the key:**



Encryption Workarounds [KS17] III

- **Try to access the PlainText without the key:**
  - **Exploit a Flaw:**
    - Weakness in the algorithm (more on that later),
    - weakness in the random-number generator (more on that later),
    - weakness in the implementation,
    - bugs (eg. Gordon's exploit on android in 2015[6]),
    - backdoors (eg. NSA NOBUS -Bullrun program- Dual EC-DRBG [BLN15]
  - **Access PlainText when in use:**
    - Access live system memory,
    - especially useful against Full Disk Encryption,
    - Seize device while in use,
    - remotely hack the device,
    - "Network Investigative Technique" (eg. Playpen case against tor).

- **Locate a PlainText copy:**
  - Avoid encryption entirely,
  - cloud providers (eg. emails),
  - remote cloud storage (eg. iCloud),

Takeaways:

- **No workaround works every time:** the fact that a target used encryption does not mean that the investigation is over.
- **some workarounds are expensive:** exploiting.
- **expertise may be have to be found outside of the governments:** vendors' assistance?

Technically, we can retain that crypto-systems have weaknesses:

- key generation,
- key length,
- key distribution,
- key storage,
- how users enter keys into the crypto-system,
- weakness in the algorithm itself / implementation,
- system / computer running the algorithm,
- crypto system used in different points in time,
- **users.**

[5] https://hashcat.net/hashcat/
[6] https://cve.circl.lu/cve/CVE-2015-3860

When cryptography helps investigations

- crypto provides authentication mechanisms.
-
-
-

Hands-on: Understanding RSA

With only one key

Several potential weaknesses:

- Key size too small: keys up to 1024 bits are breakable given the right means,
- close p and q,
- unsafe primes, smooth primes,
- broken primes (FactorDB, Debian OpenSSL bug).

Several potential weaknesses:

- share moduli: if $n_1 = n_2$ then the keys share p and q,
- share p or q,

In both case, it is trivial to recover the private keys.

Hands-on: Exploiting Weaknesses in RSA

Using Sage

Breaking small keys[7]

- Go into:

  ```
  ~/smallKey
  ```

- what is the key size of smallkey?
- what is n?
- what is the public exponent?
- what is n in base10?
- what are p and q?

Let's generate the private key: using p, then using q.

[7] https://www.sjoerdlangkemper.nl/2019/06/19/attacking-rsa/

- Go into:

  ~/ClosePQ

- use Fermat Algorithm[8] to find **both p and q:**

```
def fermatfactor(N):
  if N <= 0: return [N]
  if is_even(N): return [2,N/2]
  a = ceil(sqrt(N))
  while not is_square(a^2-N):
    a = a + 1
  b = sqrt(a^2-N)
  return [a - b,a + b]
```

[8] http://facthacks.cr.yp.to/fermat.html

Shared prime factors

Hands-on: Exploiting Weaknesses in RSA
– at bigger scale –

Snake Oil Crypto[9] - Problem Statement

IoT devices **are often the weakest devices** on a network:

- Usually the result of cheap engineering,
- sloppy patching cycles,
- sometimes forgotten–not monitored,
- few hardening features enabled.

We feel a bit safer when they use TLS, but should we?

[9] https://github.com/d4-project/snake-oil-crypto

Keep a log of links between:

- x509 certificates,
- ports,
- IP address,
- client (ja3),
- server (ja3s),

  "JA3 is a method for creating SSL/TLS client fingerprints that should be easy to produce on any platform and can be easily shared for threat intelligence."[10]

Pivot on additional data points during Incident Response

[10] https://github.com/salesforce/ja3

Snake Oil Crypto - Objectives

Collect and **store** x509 certificates and TLS sessions:

- Public keys type and size,
- moduli and public exponents,
- curves parameters.

Detect anti patterns in crypto:

- Moduli that share one prime factor,
- Moduli that share both prime factors, or private exponents,
- Small factors,
- Nonces reuse / common preffix or suffix, etc.

Focus on low hanging fruits that appeal to attackers

Researchers have shown that several devices generated their keypairs at boot time without enough entropy[11]:

```
prng.seed(seed)
p = prng.generate_random_prime()
// prng.add_entropy()
q = prng.generate_random_prime()
n = p*q
```

Given n=pq and n’ = pq’ it is trivial to recover the shared p by computing their **Greatest Common Divisor (GCD)**, and therefore **both private keys**[12].

[11] Bernstein, Heninger, and Lange: http://facthacks.cr.yp.to/
[12] http://www.loyalty.org/~schoen/rsa/

In Snake-Oil-Crypto we compute GCD[13] between:

- between certificates having the same issuer,
- between certificates having the same subject,
- on keys collected from various sources (PassiveSSL, Certificate Transparency, shodan, censys, etc.),

“Check all the keys that we know of for vendor X”

[13] using Bernstein’s Batch GCD algorithm

Snake Oil Crypto - MISP feed

2019-11-08	**Name:** crypto-material **References:** 0 **Referenced by:** 6 **uses Object 13800 (network: x509)** **uses Object 13801 (network: x509)** **uses Object 13802 (network: x509)** **uses Object 13803 (network: x509)** **uses Object 13804 (network: x509)** **uses Object 13805 (network: x509)**		
2019-11-08	Other	**p:** text	12732045980491482532629620809854872609730718866846479950748763 99251101386987265586481573653124576541684265313376164608426942 4192867704218331356123978614869
2019-11-08	Other	**q:** text	None
2019-11-08	Other	**rsa-modulus-size:** text	1024
2019-11-08	Other	**type:** text	RSA

The MISP feed:

- **Allows** for checking automatic checking by an IDS on hashed values,
- **contains** thousands on broken keys from a dozen of vendors,
- **will be accessible upon request (info@circl.lu).**

In the future:

- **Automatic** the vendor checks by performing TF-IDF on x509's subjects,
- **automatic** vendors notification.

- ✓ sensor-d4-tls-fingerprinting [14]: **Extracts** and **fingerprints** certificates, and **computes** TLSH fuzzy hash.
- ✓ analyzer-d4-passivessl [15]: **Stores** Certificates / PK details in a PostgreSQL DB.
- snake-oil-crypto [16]: **Performs** crypto checks, push results in MISP for notification
- lookup-d4-passivessl [17]: **Exposes** the DB through a public REST API.

[14] github.com/D4-project/sensor-d4-tls-fingerprinting
[15] github.com/D4-project/analyzer-d4-passivessl
[16] github.com/D4-project/snake-oil-crypto
[17] github.com/D4-project/lookup-d4-passivessl

Leveraging OpenPGP metedata

Checking for weak crypto

Get in touch if you want to join/support the project, host a passive SSL sensor or contribute

- Collaboration can include research partnership, sharing of collected streams or improving the software.
- Contact: info@circl.lu
- https://github.com/D4-Project - https://twitter.com/d4_project

Tpm-fail: TPM meets timing and lattice attacks, 29th USENIX Security Symposium (USENIX Security 20) (Boston, MA), USENIX Association, August 2020.

Nadhem J. Al Fardan and Kenneth G. Paterson, *Lucky thirteen: Breaking the tls and dtls record protocols*, Proceedings of the 2013 IEEE Symposium on Security and Privacy (Washington, DC, USA), SP '13, IEEE Computer Society, 2013, pp. 526–540.

Ross J. Anderson, *Security engineering: A guide to building dependable distributed systems*, 2 ed., Wiley Publishing, 2008.

Jean-Philippe Aumasson, *Serious cryptography: A practical introduction to modern encryption*, no starch press, 2017.

Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen, *Dual ec: A standardized back door*, IACR Cryptology ePrint Archive **2015** (2015), 767.

References II

- Naomi Benger, Joop van de Pol, Nigel P. Smart, and Yuval Yarom, *"ooh aah... just a little bit" : A small amount of side channel can go a long way*, Cryptographic Hardware and Embedded Systems – CHES 2014 (Berlin, Heidelberg) (Lejla Batina and Matthew Robshaw, eds.), Springer Berlin Heidelberg, 2014, pp. 75–92.
- Dieter Gollmann, *Computer security (3. ed.)*, Wiley, 2011.
- Orin S. Kerr and Bruce Schneier, *Encryption workarounds*, SSRN Electronic Journal (2017).
- Kohei Matsuda, Tatsuya Fujii, Natsu Shoji, Takeshi Sugawara, Kazuo Sakiyama, Yu-Ichi Hayashi, Makoto Nagata, and Noriyuki Miura, *A 286 F2/cell distributed bulk-current sensor and secure flush code eraser against laser fault injection attack on cryptographic processor*, IEEE Journal of Solid-State Circuits **53** (2018), no. 11, 3174–3182.

References III

- Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of applied cryptography*, 1st ed., CRC Press, Inc., Boca Raton, FL, USA, 1996.
- Dag Arne Osvik, Adi Shamir, and Eran Tromer, *Cache attacks and countermeasures: The case of aes*, Topics in Cryptology – CT-RSA 2006 (Berlin, Heidelberg) (David Pointcheval, ed.), Springer Berlin Heidelberg, 2006, pp. 1–20.
- Joint Reports, *First report of the observatory function on encryption*, Tech. report, EUROPOL - EC3, 2019.
- Yuval Yarom and Katrina Falkner, *Flush+reload: A high resolution, low noise, l3 cache side-channel attack*, 23rd USENIX Security Symposium (USENIX Security 14) (San Diego, CA), USENIX Association, August 2014, pp. 719–732.