

Format Oracles on OpenPGP

F. Maury J.-R. Reinhard O. Levillain H. Gilbert

ANSSI, France

CT-RSA
April 22, 2015



Contribution

- We identify new format oracles on OpenPGP implementations of symmetrically (authenticated and) encrypted data
- We study the number of oracle requests necessary to recover a plaintext

Format oracle	Requests per byte	Affected implementation
Invalid packet tag	2	GnuPG
Double literal	2^6	GnuPG
MDC packet header	2^8	GnuPG, End-to-End

Padding Oracles

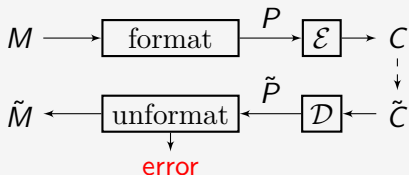
- An encryption scheme is modeled by two (inverse processes)

$$\mathcal{E} : P, K \rightarrow C$$

$$\mathcal{D} : C, K \rightarrow P \text{ (or } \perp, \text{ in authenticated encryption)}$$

- Issue:**

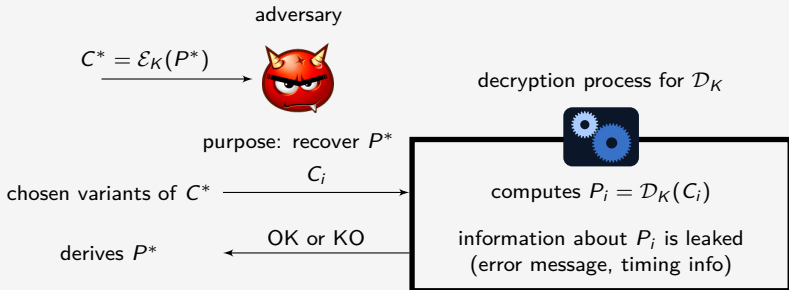
- Before encryption, the plaintext is usually prepared following a specific format, e.g., a padding is applied
- After decryption, this format has to be removed. This process may raise errors if the format is not followed



- The presence/absence of error leaks information on the result of decryption
- Using malleability of \mathcal{D} , this leakage can be aggregated to decipher a target ciphertext

Padding/Format Oracles Attacks

- This principle can be used to mount chosen ciphertext attacks enabling to decipher a target ciphertext



- Previous results

- Bleichenbacher on RSA-PKCS#1v1.5, in SSL/TLS [BI98]
- Vaudenay on CBC mode used with specific padding schemes, in SSL or IPsec [Va02]
- Klíma and Rosa noted that the format has not to be restricted to cryptographic padding, but may be applicative [KR03]

OpenPGP

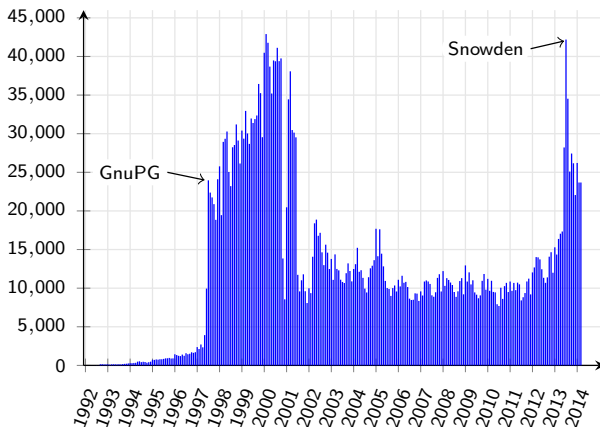
OpenPGP

- Pretty Good Privacy: published by P. Zimmermann in 1991
- Application enabling to protect
 - Confidentiality e.g. of emails, through hybrid encryption
 - Authenticity, through signature
- Standardized by IETF from 1997: OpenPGP message format
 - RFC 2440, november 1998
 - Updated by RFC 4880, november 2007
- Main free implementation of the standard: GnuPG
- Renewed interest following the Snowden revelations
 - Increase in the number of monthly registered public keys
 - Multiple promotional campaigns

OpenPGP

- Pre
- App
- Sta
- Ma
- Res

Number of keys registered (by month)



OpenPGP

- Pretty Good Privacy: published by P. Zimmermann in 1991
- Application enabling to protect
 - Confidentiality e.g. of emails, through hybrid encryption
 - Authenticity, through signature
- Standardized by IETF from 1997: OpenPGP message format
 - RFC 2440, november 1998
 - Updated by RFC 4880, november 2007
- Main free implementation of the standard: GnuPG
- Renewed interest following the Snowden revelations
 - Increase in the number of monthly registered public keys
 - Multiple promotional campaigns

OpenPGP

Promotion of GnuPG by FSF Europe

Email self-defense against surveillance

Most surveillance relies on fundamental aspects of a message in its format of content.
But we can defend ourselves.



The Problem

The assumption that private communication is safe is false. In private communication, the message is sent in plain text. Each email sent over the internet is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network.


Encryption

Each time you press the 'send' button, a message is sent to the server. The message is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network.

The Solution


Instead of sending a message in plain text, you can send it encrypted. The message is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network. The message is sent in plain text. It is intercepted by anyone who can access the network.

Private email communication



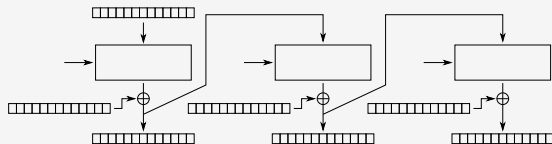
Take back your privacy! Use GnuPG!

- Free Software
- for all email addresses
- for GNU/Linux, Windows, Mac, Android, ...
- no account or registration needed
- free of charge



OpenPGP Encryption Mode

- Symmetric encryption is done in OpenPGP with CFB mode



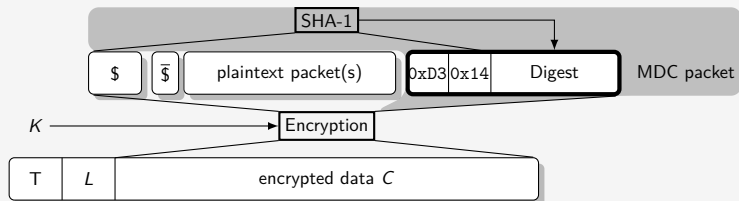
- CFB is used with an all zero IV, and is made non-deterministic by prepending a random block to the plaintext
- The first 2 bytes of the initial random block are repeated
 - This provides a quick consistency check at the beginning of decryption, useful for password based encryption
 - This check has been used by [MZ05] to decipher 2 bytes per block with an oracle attack
- No padding, truncation of the keystream
- Authenticated encryption uses an ad-hoc mode
 - Security?

OpenPGP Message Format

■ Packet Structure



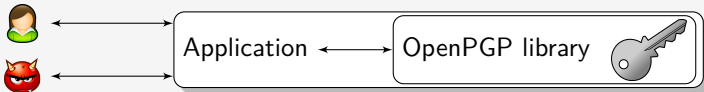
■ Encrypted Packet (with Integrity Protection)



Oracles

Format Oracles in OpenPGP Implementations

- We investigated implementations that are (or can be seen) as libraries, to use to develop higher-level applications relying on OpenPGP



- We expect these implementations to act as cryptographic back ends for the front end applications:
 - Perform all cryptographic operations
 - As a consequence, be the only part where keys are manipulated
- **Issue:** The interaction between application and library often goes beyond the ideal model of encryption schemes
 - Error messages are output (or logged)
 - The API of the library does not state whether these errors are sensitive
 - There is a risk that the front end may leak them
- **Result:** Identification of 3 types of leakage, potential oracles (over 50 distinct oracles)

Format Oracles in OpenPGP Implementations

- We investigated implementations that are (or can be seen) as libraries, to use to develop higher-level applications relying on OpenPGP



Application

OpenPGP library

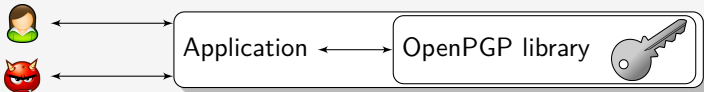


Investigated implementations

- GnuPG, originally an application, but can be used as a library through scripting, even produces *status messages* for the calling application for such cases.
- End-to-End, Google OpenPGP implementation in JavaScript
- OpenPGP.js, another library developed in JavaScript
- Error messages are output (or logged)
- The API of the library does not state whether these errors are sensitive
- There is a risk that the front end may leak them
- **Result:** Identification of 3 types of leakage, potential oracles (over 50 distinct oracles)

Format Oracles in OpenPGP Implementations

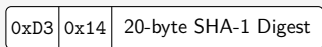
- We investigated implementations that are (or can be seen) as libraries, to use to develop higher-level applications relying on OpenPGP



- We expect these implementations to act as cryptographic back ends for the front end applications:
 - Perform all cryptographic operations
 - As a consequence, be the only part where keys are manipulated
- **Issue:** The interaction between application and library often goes beyond the ideal model of encryption schemes
 - Error messages are output (or logged)
 - The API of the library does not state whether these errors are sensitive
 - There is a risk that the front end may leak them
- **Result:** Identification of 3 types of leakage, potential oracles (over 50 distinct oracles)

MDC Packet Header Oracle

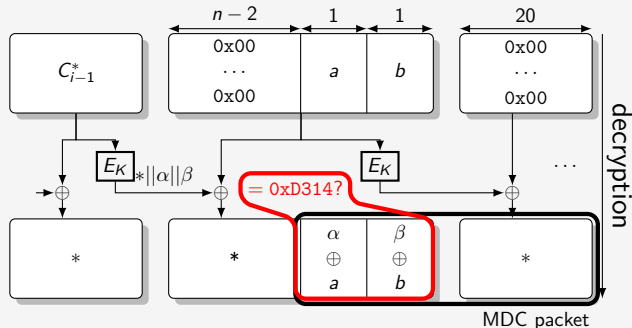
- For integrity-protected encrypted packets, the last 22 bytes of the decrypted ciphertext form a Modification Detection Code packet



- GnuPG and End-to-End enforced this format by specifically checking for the two byte values 0xD314 at position 22 and 21 from the end of the decrypted ciphertext, and returning specific error messages in case of mismatch
- Using this leakage and CFB malleability, it is possible to recover any two bytes of plaintext by performing 2^{16} oracle queries

MDC Packet Header Oracle Attack

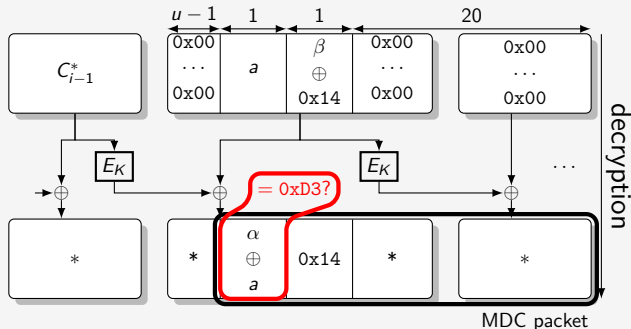
- $P_i^* = C_i^* \oplus E_K(C_{i-1}^*)$, recover $P_i^* \Leftrightarrow$ recover $E_K(C_{i-1}^*)$
- Recovering the last 2 bytes of $E_K(C_{i-1}^*)$



- For all possible byte pairs (a, b) , build and submit ciphertext $C_{a,b}$, with a and b located at the position of the MDC packet header
- From the only value that does not raise a MDC format error, deduce 2 bytes of $E_K(C_i^*)$
- Complexity: at most 2^{16} requests

MDC Packet Header Oracle Attack

- Additional bytes of $E_K(C_{i-1}^*)$ can be recovered for 2^8 requests per byte
- **Idea:** tweak the ciphertexts to ensure one of the 2 byte conditions



- It is possible to avoid the 2^{16} initial search by using more advanced techniques
- Final complexity: for messages over 4kB, 2^8 requests per byte [detail](#)

Invalid Identifier Oracles and Double Literal Oracle

- After decryption, the plaintext is an OpenPGP message, and is parsed by the OpenPGP implementation
- During this parsing, errors may be encountered, for example:
 - An identifier value (tag, algorithm identifier, ...) is invalid
 - There are two literal packets
- All the libraries raise some sort of error in one case or another
 - For example, GnuPG emits a status message when confronted with two consecutive literal packets.
- Using a tag oracle, it is possible to recover an arbitrary byte for 2^8 requests
- **Idea:** submit ciphertexts that decrypt into 2 consecutive packets, with the tag of the second packet located at the target byte position

Impact

Downgrade attacks

- CFB mode is used in all the encryption contexts
- A same key can be reused independently of the context
- It is possible to decrypt any type of OpenPGP encrypted data with any OpenPGP format oracle

Application

- Usual application: email protection
 - Disconnected mode: difficult to get error feedback
 - Key unlocking: user interaction may be necessary
- However,
 - OpenPGP is used in a lot of contexts, e.g. chat
 - The use of OpenPGP MUA proxies is considered, which might introduce unattended decryption oracles, with a feedback to the attacker

Conclusion

Disclosure

- We informed the affected libraries developers on our results early on
- GnuPG and End-to-End patched the MDC packet header oracle
- Varying stance relatively to the other oracles:
 - End-to-End and OpenPGP.js propose a high-level API, whose errors are sanitized
 - GnuPG considers it is the responsibility of front end developers not to mishandle the errors. They documented the sensitivity of these messages

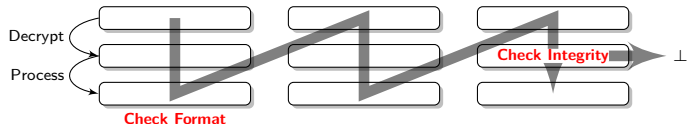
How To Prevent Format Oracles?

- After more than 15 years of format oracles, it is still possible to uncover such “flaws” in major cryptographic implementations
- **Solution:** authenticated encryption
 - Mandating authenticated encryption is a systematic way to avoid format oracles
- **Warning:** implementation robustness
 - As illustrated by the MDC packet header oracle, use of authenticated encryption is not sufficient
 - Implementations have to perform decryption steps in the right order
 - Counter example: GnuPG implementation

How To Prevent Format Oracles?

- After more than 15 years of format oracles, it is still possible to uncover such “flaws” in major cryptographic implementations
- **Solution:** authenticated encryption

GnuPG implementation



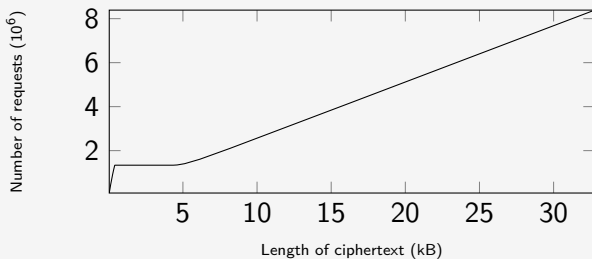
- Counter example: GnuPG implementation

How To Prevent Format Oracles?

- After more than 15 years of format oracles, it is still possible to uncover such “flaws” in major cryptographic implementations
- **Solution:** authenticated encryption
 - Mandating authenticated encryption is a systematic way to avoid format oracles
- **Warning:** implementation robustness
 - As illustrated by the MDC packet header oracle, use of authenticated encryption is not sufficient
 - Implementations have to perform decryption steps in the right order
 - Counter example: GnuPG implementation
 - Adopt Decrypt-Verify-Then-Release, requires buffered decryption
 - Intermediate integrity tags if buffered decryption is not acceptable

Thank you for your attention

MDC Packet Header Attack Complexity



- Number of requests necessary to decipher a ciphertext
 - For short messages, the advanced strategy does not gain anything
 - For messages of intermediate length, it is useful, but it entails a fixed cost
 - For long messages, it can be applied for free