Wombat: one more Bleichenbacher attack toolkit

Olivier Levillain   Aina Toky Rasoamanana

Télécom SudParis

GreHack 2019
15 novembre 2019

Plan

RSA and PKCS#1 v1.5 in a nutshell

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Current results and future work

Conclusion

Plan

RSA 101

RSA

- a pervasive cryptosystem
- asymmetric encryption and signature

RSA 101

RSA

- a pervasive cryptosystem
- asymmetric encryption and signature

Details

- public key $n = p \cdot q$, e
- private key d
- raw encryption : $C = M^e[n]$
- raw decryption : $C^d = M^{ed} = M[n]$

RSA 101

RSA

- a pervasive cryptosystem
- asymmetric encryption and signature

Details

- public key $n = p \cdot q$, e
- private key d
- raw encryption : $C = M^e[n]$
- raw decryption : $C^d = M^{ed} = M[n]$

Problems with raw RSA operations

- if e and M are *small*
- malleability w.r.t. the multiplication

The need for a padding scheme

We thus need to format the message before encrypting (or signing) it

- PKCS#1 standardize how to use RSA
- in particular, the document defines different padding scheme

The need for a padding scheme

We thus need to format the message before encrypting (or signing) it

- PKCS#1 standardize how to use RSA
- in particular, the document defines different padding scheme

In this talk, we are most interested in padding type 2, described in version 1.5 of the standard, and used for encryption :

$\longleftarrow$ n bytes $\longrightarrow$

00	02	non zero random bytes	00	Encapsulated data

$\longleftarrow$ 8+ bytes $\longrightarrow$

Other padding schemes

PKCS#1 v1.5 also describes two other schemes, which are deterministic

- padding type 0 (zero bytes, rarely used)
- padding type 1 (`ff` bytes, used for signature)

PKCS#1 v2.1

- OAEP (Optimal Asymmetric Encryption Padding) for encryption
- PSS (Probabilistic Signature Scheme) for signature
- these schemes have better security properties...
- ... but are not always used in standards

Plan

An observation about padding type 2

$\longleftarrow$ n bytes $\longrightarrow$

00	02	non zero random bytes	00	Encapsulated data

$\longleftarrow$ 8+ bytes $\longrightarrow$

An observation about padding type 2

$\longleftarrow$ n bytes $\longrightarrow$

00	02	non zero random bytes	00	Encapsulated data
		$\longleftarrow$ 8+ bytes $\longrightarrow$		

With a correctly formated message to be encrypted

- the *raw* plaintext M starts with 00 02
- interpreted as an integer, this means that M is an integer between $2B$ and $3B$
  - with $B = 2^{(|n|-16)}$
  - where $|n|$ is the size of the modulus n in bits

Attack principle (CRYPTO 1998)

We assume there exists an oracle which

- accepts to decrypt messages
- returns true when the padding was correct, false otherwise
- (the decrypted message will be kept secret)

Attack principle (CRYPTO 1998)

We assume there exists an oracle which

- accepts to decrypt messages
- returns true when the padding was correct, false otherwise
- (the decrypted message will be kept secret)

An attacker wishing to recover $m = c^d$ can then

- send altered messages $c \cdot s^e$ (with s known)
- let the server handle $(c \cdot s^e)^d = c^d \cdot s^{ed} = ms$
- infer that $2B \leq ms < 3B$ in case the oracle returns true
- repeat the operations, and recover m with these equations
- (this is an adaptive chosen ciphertext attack)

Different oracle types (1/2)

In practice, the attacker wants to find messages starting with 00 02

Different oracle types (1/2)

In practice, the attacker wants to find messages starting with 00 02

$\longleftarrow$ n bytes $\longrightarrow$

00	02	non zero random bytes	00	Encapsulated data

$\longleftarrow$ 8+ bytes $\longrightarrow$

However, some oracles also make additional checks

- the padding contains at least 8 bytes
- the padding ends with a null byte
- the message obtained has the expected length

Different oracle types (2/2)

If we assume an oracle returning true only for messages

- starting with 00 02
- where the padding contains at least 8 bytes
- and where the padding ends

The attacker thus loses *good* messages (starting with 00 02) which would have led to interesting equations

Different oracle types (2/2)

If we assume an oracle returning true only for messages

- starting with 00 02
- where the padding contains at least 8 bytes
- and where the padding ends

The attacker thus loses *good* messages (starting with 00 02) which would have led to interesting equations

Bardou et al. proposed a classification where each oracle type depends on the messages an attacker can distinguish

Results from Bardou et al.

The article, published at CRYPTO 2012, improved the original algorithms (CRYPTO 1998)

Oracle type	Average nb of requests	
	Original algo	Improved algo
FFF	-	18 040 221
FFT	215 982	49 001
FTT	159 334	39 649
TFT	39 536	10 295
TTT	38 625	9 374

Plan

RSA and PKCS#1 v1.5 in a nutshell

Bleichenbacher : *the million-message attack*

Wombat : one more Bleichenbacher toolkit

Current results and future work

Conclusion

A modular way to implement the attack (1/2)

To test an implementation, we write a *stub*, which allows

- to get the RSA public key
- to get a challenge (an encrypted message)
- to submit messages to be decrypted

A modular way to implement the attack (1/2)

To test an implementation, we write a *stub*, which allows

- to get the RSA public key
- to get a challenge (an encrypted message)
- to submit messages to be decrypted

The attacker can submit messages for which the plaintext is known, and assess the oracle type

- well formed messages
- messages not starting with 00 02
- messages with a short padding
- messages with an unending padding

A modular way to implement the attack (2/2)

If the attacker can identify *good* messages by observing the implementation behaviour, an oracle has been identified

The attacker can then

- evaluate more precisely the cost of the attack
- attack the implementation to recover the plaintext corresponding to the challenge
- use the oracle to forge a signature

A modular way to implement the attack (2/2)

If the attacker can identify *good* messages by observing the implementation behaviour, an oracle has been identified

The attacker can then

- evaluate more precisely the cost of the attack
- attack the implementation to recover the plaintext corresponding to the challenge
- use the oracle to forge a signature

`wombat` currently implements

- the original attack from Daniel Bleichenbacher
- improved versions of the attack (Bardou et al.)
- pure oracles to validate the attacks
- a TLS *stub* TLS (more on this later)

An open-source framework

Wombat

- tool developed in Python during an internship
- version 0.1 published in September
- `https://gitlab.com/pictyeye/wombat`

An open-source framework

Wombat

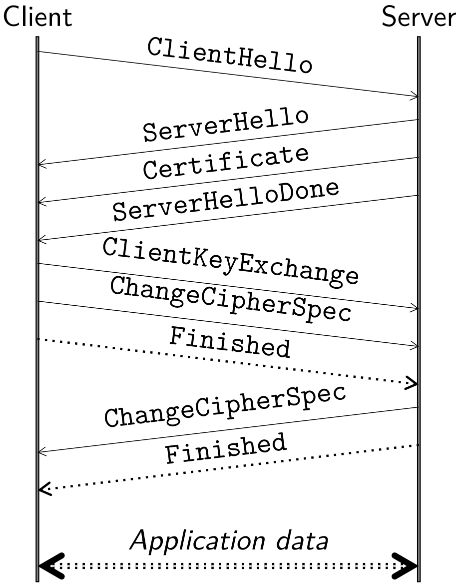
- tool developed in Python during an internship
- version 0.1 published in September
- `https://gitlab.com/pictyeye/wombat`

Possible usages

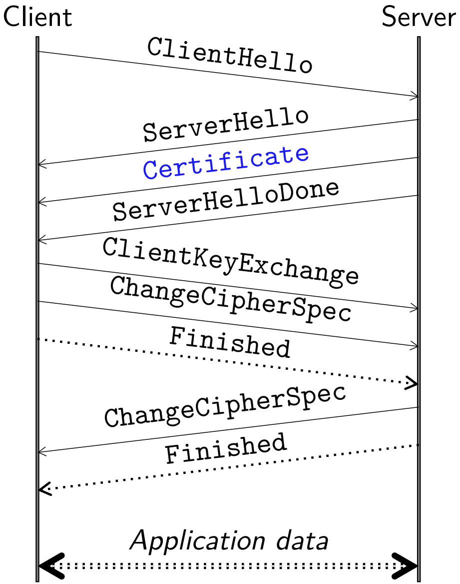
- help identifying existing oracles
- mount attacks (with respect to the law and morality)
- mount hands-on sessions in classes (a simple TCP server is provided)

Plan

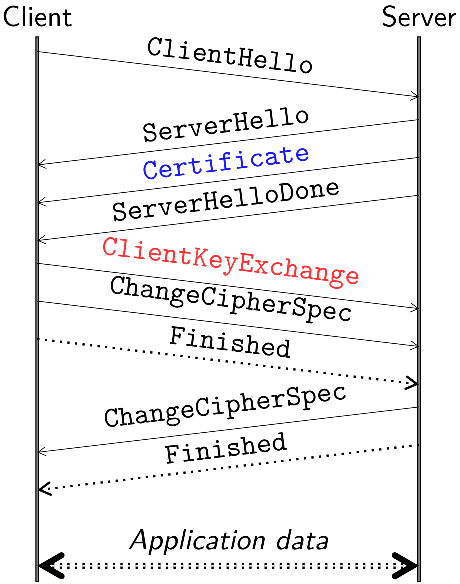
Application to TLS (1/3)



Application to TLS (1/3)



Application to TLS (1/3)



Application to TLS (2/3)

We wrote a *stub*

- public key recovery from the server certificate
- creation of a target ciphertext (the challenge) including a *pre-master-secret*
- interface to send message to be decrypted to the TLS server
    - first messages to force the use of RSA key exchange
    - emission of the `ClientKeyExchange` including the encrypted message to test
    - observation of the server reaction (received messages, delay before an answer)

Identification and exploitation of an oracle

Application to TLS (3/3)

We added the vulnerability to `mbedtls` (the actual implementation is very robust, including against timing attacks)

```
+   // DON'T DO THIS AT HOME
+   if (ret == MBEDTLS_ERR_RSA_INVALID_PADDING) {
+       mbedtls_ssl_send_alert_message( ssl, MBEDTLS_SSL_ALERT_LEVEL_FATAL,
+                                       MBEDTLS_SSL_ALERT_MSG_DECRYPT_ERROR );
+       return (MBEDTLS_ERR_SSL_BAD_HS_CLIENT_KEY_EXCHANGE);
+   }
```

Application to TLS (3/3)

We added the vulnerability to `mbedtls` (the actual implementation is very robust, including against timing attacks)

```
+   // DON'T DO THIS AT HOME
+   if (ret == MBEDTLS_ERR_RSA_INVALID_PADDING) {
+       mbedtls_ssl_send_alert_message( ssl, MBEDTLS_SSL_ALERT_LEVEL_FATAL,
+                                       MBEDTLS_SSL_ALERT_MSG_DECRYPT_ERROR );
+       return (MBEDTLS_ERR_SSL_BAD_HS_CLIENT_KEY_EXCHANGE);
+   }
```

Demo time

Real-life tests on TLS

We believed that explicit Bleichenbacher oracles in TLS stacks were something from the past...

Real-life tests on TLS

We believed that explicit Bleichenbacher oracles in TLS stacks were something from the past...

```
% python example_BB_TLS_prober.py <tested-server>
Explicit oracle found (type FFT)
   true-signal: set([(False, 21, 51), TLSHandshakeFailed()])
   false-signal: set([(False, None, None), (False, 21, 20)])
```

Real-life tests on TLS

We believed that explicit Bleichenbacher oracles in TLS stacks were something from the past...

```
% python example_BB_TLS_prober.py <tested-server>
Explicit oracle found (type FFT)
   true-signal: set([(False, 21, 51), TLSHandshakeFailed()])
   false-signal: set([(False, None, None), (False, 21, 20)])
```

And interestingly, `<tested-server>` is from Top Alexa 1M !

Future work on TLS

Improve the TLS prober to be more reliable and more precise

- improve the interpretation of timing distributions
- handle more messages from the server behaviour (TCP errors for example)
- use different message sequences to trigger a reaction from the server

Future work on TLS

Improve the TLS prober to be more reliable and more precise

- improve the interpretation of timing distributions
- handle more messages from the server behaviour (TCP errors for example)
- use different message sequences to trigger a reaction from the server

Launch campaigns to *identify* potential oracles in a more systematic way

- regular HTTPS Top Alexa 1M scans
- SMTP servers (where we usually find obsolete software)

Future work on TLS

Improve the TLS prober to be more reliable and more precise

- improve the interpretation of timing distributions
- handle more messages from the server behaviour (TCP errors for example)
- use different message sequences to trigger a reaction from the server

Launch campaigns to *identify* potential oracles in a more systematic way

- regular HTTPS Top Alexa 1M scans
- SMTP servers (where we usually find obsolete software)

Implement more sophisticated attacks such as DROWN or TLS 1.3/QUIC signature forgery

Other applications

PKCS#1 is present in other standards

- XML Encryption
- SSH (RFC 4432)
- OpenPGP

Other applications

PKCS#1 is present in other standards

- XML Encryption (proposed to students)
- SSH (RFC 4432)
- OpenPGP

Other applications

PKCS#1 is present in other standards

- XML Encryption (proposed to students)
- ~~SSH~~ (RFC 4432 uses OAEP)
- OpenPGP

Preliminary results on gpg

Setup

- an RSA key used for encryption
- an encrypted message
- altered versions of the encrypted messages

Preliminary results on gpg

Setup

- an RSA key used for encryption
- an encrypted message
- altered versions of the encrypted messages

Decryption time...

Preliminary results on gpg

Setup

- an RSA key used for encryption
- an encrypted message
- altered versions of the encrypted messages

Decryption time...

- correct format : `Invalid cipher algorithm` or decryption
- padding too short : `Invalid cipher algorithm`
- invalid first bytes : `Wrong secret key used`
- message is only padding : `Wrong secret key used`

It is an FTT oracle ! Hopefully it is rare to be able to submit encrypted files to OpenPGP and observe the error messages.

Plan

Conclusion

The so-called *million-message attack* from Bleichenbacher

- an attack well known for a long time
- a non trivial example to teach cryptographic attacks
- still a reality today ?

Wombat

- an open source tool to test the attack
- helpful to reproduce existing attacks
- extensible via *stubs* to analyse other standards
- useful for hands-on sessions with students

Questions ?

Thank you for your attention

https://paperstreet.picty.org/yeye

https://gitlab.com/pictyeye/wombat