Dinu Alexandru
342C4

Cryptography Project

Bleichenbacher Chosen Ciphertext Attack on PKCS #1 v1.5

For this project, I have implemented Daniel Bleichenbacher's Adaptive Chosen Ciphertext Attack on RSA PKCS #1 v1.5 [1]. It is a particularly interesting attack due to real life implications (PKCS #1 was widely deployed, for example, in HTTPS – SSL/TLS) and due to the mathematics behind it. This specific attack exploits the implementation flaws found in numerous servers, that is, they report whether the encoding of the message they have received is PKCS1-conforming. This allows the attacker to use the server as an oracle to which he sends several queries. Based upon the server's reply, the attacker can gain information about the complete decryption of an intercepted ciphertext (hence the *adaptive chosen ciphertext* bit).

PKCS #1 v1.5 encoding scheme

The need for standardization led to the implementation of a relatively large number of various encoding schemes for messages [2]. In this project, as stated in the overview above, I am focusing on PKCS #1 v1.5. The encoding of a message M using this scheme looks as follows:

$$PKCS1(M) = 0x00 \mid 0x02 \mid [non - zero\ padding\ bytes] \mid 0x00 \mid [M]$$

The first 2 bytes are constant (0x00, 0x02) and they define the mode of operation (here, encryption). Next, there are at least 8 non-zero padding bytes, followed by the constant separation byte 0x00, and finally our message. Using this scheme, the message length in bytes must not exceed k – 11 (where k is the total size of the encoded message = size of RSA modulus, in bytes).

Therefore, under this scheme, a server[1] that receives a ciphertext c (RSA encrypted), decrypts it and checks whether the first 2 bytes of the plaintext are indeed [0x00, 0x02]. If this is not the case, then the server reports an error message to the client (e.g. Invalid Encoding).

[1] The term *server* is used interchangeably with the term *oracle*.

Assume that the attacker has intercepted a ciphertext and he wants its decryption. He can exploit the behaviour of the server (using it as an oracle), in the following way:

$$\text{oracle}(c) = \text{True} \equiv \left(c^d \ (\text{mod } n)\right)[0{:}2] == \backslash x00\backslash x02$$

where $c^d(mod\ n)$ is the RSA decryption of the ciphertext (performed by the server).

In his paper [1], Daniel Bleichenbacher showed that roughly 2^{20} carefully chosen ciphertexts are needed in order to completely decrypt the message (although this number may vary depending on numerous implementation details).

RSA and PKCS #1 v1.5

For this project, RSA cryptosystem is used, therefore, I have also implemented a simple RSA key generation algorithm. The numbers are kept quite small (for this proof-of-concept): modulus (n) size is 256 bits (p, q are 128-bit primes), public exponent e = 3. Each prime number is generated using random search in the 128-bit range. Also, it is compulsory that they differ and that none of them are a multiple of e. Afterwards, ϕ is computed as the product $\phi = (p-1) \cdot (q-1)$, and then the private exponent $d \equiv e^{-1}(mod\ \phi)$, that is, d is the modular inverse of e, with respect to ϕ.

In PKCS #1, the size of the encoded message is the same as the modulus size. The complete encryption algorithm, under this scheme, is the following: first, the message is encoded using PKCS #1; then, the result is converted to an integer value, encrypted using RSA, and then converted back to a bytestring. The oracle (i.e. server) receives the encrypted bytestring value, converts it back to an integer, applies RSA decryption using (n, d), converts the result to a bytestring, and then checks whether the first 2 bytes are \x00\x02.

Dinu Alexandru
342C4

Attack

As described in the paper [1], the attack can be divided into three phases. Simply speaking, Bleichenbacher's algorithm is the following:

Let c denote the ciphertext that the attacker has intercepted. To construct a query, generate a value s and send $c' = cs^e (mod\ n)$ - this corresponds to multiplying the plain message m by s, because $cs^e (mod\ n) = (ms)^e (mod\ n)$.

After decryption, the server obtains $m' = (cs^e)^d (mod\ n)$ and reports whether the first 2 bytes of the bytestring conversion of m' are 0x00 and 0x02, respectively. If a random message m is sent to the server, the probability that m is PKCS conforming is roughly in the interval $[2^{-15}, 2^{-17}]$. The exact probability highly depends on the size and value of the RSA modulus n.

The value s can be found with a non-negligible probability. After finding the value of s such that the ciphertext decrypts to a proper PKCS encoded message, the attacker will know that for this value of s, $ms\ (mod\ n)$ is in a specific range (i.e. the range of integers which begins with \x00\x02 when converted to bytestring, using big-endian representation).

The attack is said to be adaptive in the sense that future queries are constructed based upon the information obtained from the previous server replies. Thus, the rest of the attack sends queries with carefully chosen values of s and narrows the range of values it can take, up to a point when an interval of the form $[a, a]$ is found. Finally, the bytestring value of a is the plaintext that we are interested in.

Formal description

As stated above, there are three phases of the attack (four if we also account for the blinding step, but in this implementation, it is assumed that the attacker has intercepted the ciphertext of a PKCS #1 encoded message, so blinding is not necessary).

Let $B = 2^{8(k-2)}$ be the length of the message, in bits, without the first 2 bytes; k is the length of the RSA modulus, in bytes (256 / 8 = 32, in this implementation). Since ms is PKCS conforming: $2B \leq ms(mod\ n) < 3B$. Let $M = \{[2 \cdot B, 3 \cdot B - 1]\}$ be the initial set of intervals (the interval represents the broadest range of possible s-values).

Dinu Alexandru
342C4

1. Searching

We start the search by trying to find the smallest $s_1 \geq \frac{n}{3 \cdot B}$, such that $cs_1^e (mod\, n)$ is PKCS conforming. Next, we continue the search based upon the size of M (i.e. the number of intervals in M).

If M contains at least 2 intervals, then look for the smallest $s_i \geq s_{i-1}$ such that $cs_\text{i}^e (mod\, n)$ is PKCS conforming. Otherwise, if M contains exactly one interval of the form $[a, b]$, then use the previously calculated s-value to derive lower and upper bounds for the next s-value, that is, choose:

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n}$$

$$\frac{2B + r_i n}{b} \leq s_i < \frac{3B + r_i n}{a}$$

until we arrive at a PKCS conforming ciphertext $cs_\text{i}^e (mod\, n)$.

2. Narrowing the set of solutions

After finding the new value s_i, update the set M of intervals as follows:

$$M_i \leftarrow \bigcup_{(a,b,r)} \left\{ \left[\max\left(a, \left\lceil \frac{2B + rn}{s_i} \right\rceil \right), \min\left(b, \left\lfloor \frac{3B - 1 + rn}{s_i} \right\rfloor \right) \right] \right\}$$

$$\text{for all } [a, b] \in M_{i-1} \text{ and } \frac{as_i - 3B + 1}{n} \leq r \leq \frac{bs_i - 2B}{n}.$$

3. Checking for solution

If M only contains one interval of the form $[a, a]$, then we have reached the solution. $m \leftarrow a\ (mod\, n)$ is the complete decryption of the intercepted ciphertext.

Otherwise, repeat the steps above.

Dinu Alexandru
342C4

Analysis

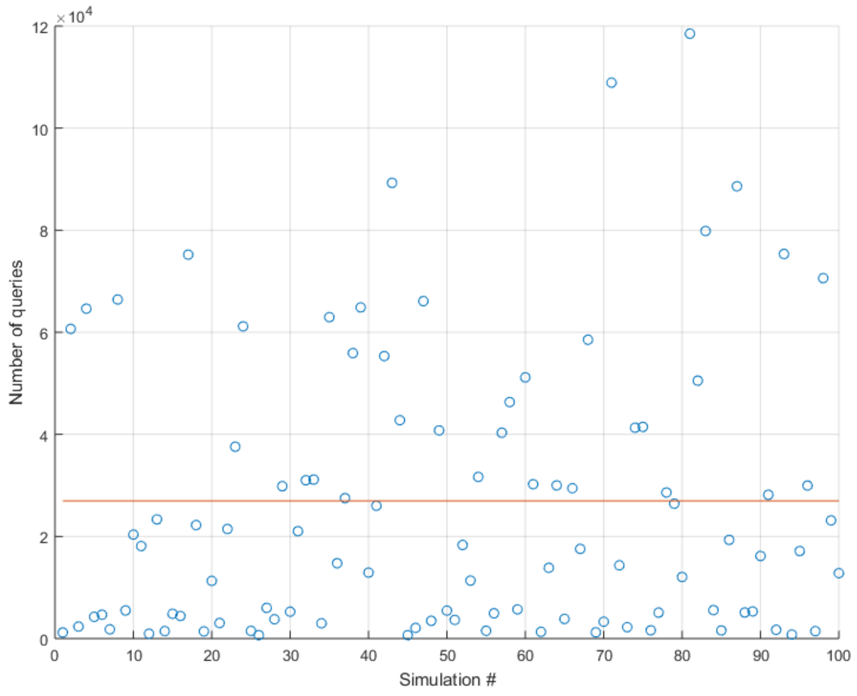
In Section 3.2 of the paper [1], Daniel Bleichenbacher thoroughly analyses the correctness of the attack and approximates the complexity, so I will not go into a high level of detail. However, it is worth mentioning some key points:

- the derivation of the bounds of the intervals of M (described above) is done to minimize the number of queries and speed up the search
- the length of an interval in M_i is upper bounded by $\left\lceil \frac{B}{s_i} \right\rceil$
- the searching algorithm when there is only one interval in M takes almost logarithmic time
- the remaining interval in M (for the case above) is divided in half each time that step occurs (almost every iteration)
- there are very few cases when the size of $M \geq 2$ (in theory, it is expected that the search part where there are at least 2 intervals in M to be executed only once)

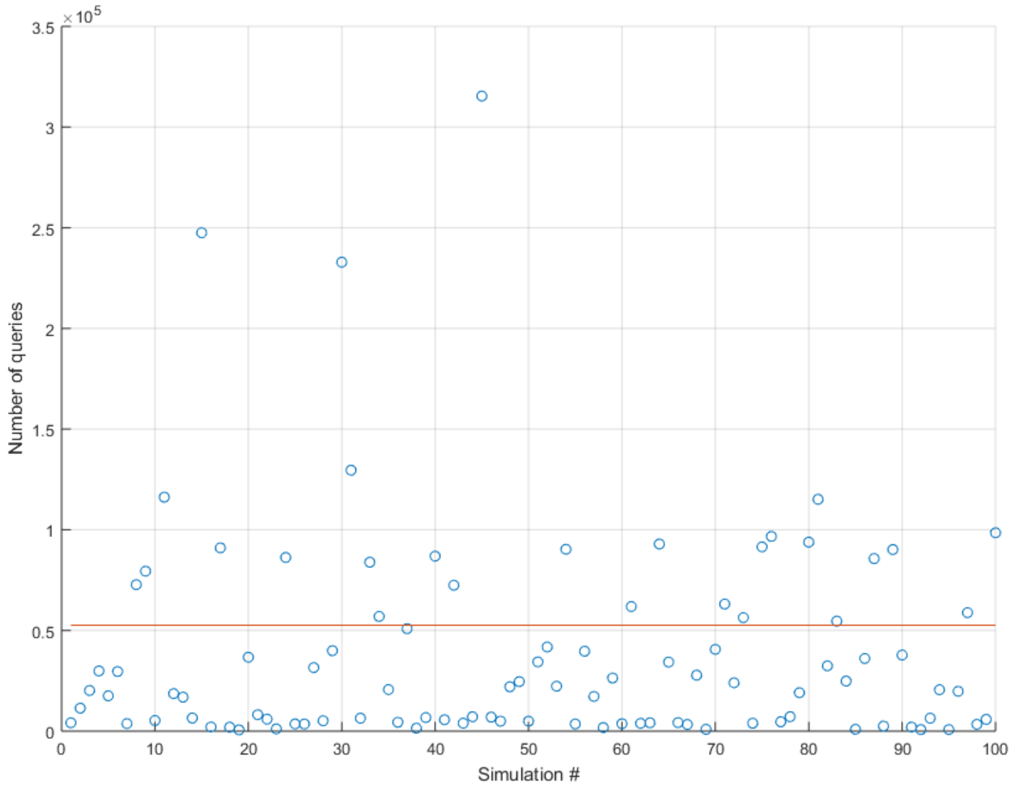
My results:

The setup is a 256-bit long RSA modulus (p, q are 128-bit primes).

For a constant 11-byte message, the average number of queries for 100 *simulations* is 25605.92, but the standard deviation is 26985 which is because of the random component. The minimum number of queries was 667, and the maximum number was 118511.



For a random 11-byte message for each one of the 100 simulations, the results are the following: mean = 38587, standard deviation = 52598, minimum number of queries = 603, maximum number of queries = 315410.



Conclusions

To conclude, I have to say that Bleichenbacher’s attack is a very interesting and didactic application of real-life cryptography that I enjoyed implementing. The popularity of the PKCS #1 v1.5 (e.g. used in HTTPS) and some flawed implementations of the verification of correctness of this encoding scheme make a perfect candidate for a highly exploitable setup.

References

[1] Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1462*, 1–12. https://doi.org/10.1007/BFb0055716

[2] https://en.wikipedia.org/wiki/PKCS