Notes on applied cryptography

Gianluca Dini

November 8, 2007

1 The RSA Algorithm

1.1 Small exponent attack

Let us consider a system in which a group of entities have all the same encyption exponent e, but, each entity has a distinct modulus. If an entity A wishes to send the same message m to three entities whose public moduli are n_1, n_2, and n_3, and whose encryption exponents are $e = 3$, then A would send $c_i = m \bmod n_i, i = 1, 2, 3$. Notice that by definition it must be $m < n_i, i = 1, 2, 3$, and thus $m^3 < n_1 n_2 n_3$. Furthermore, if moduli are pairwise prime (which is very likely), an eavesdropper[1] observing c_1, c_2, and c_3 can apply the CRT (using the Gauss's algorithm) to find a solution x, $0 \leq x \leq n_1 n_2 n_3$, to

$$\begin{cases} x \equiv c_1 \pmod{n_1} \\ x \equiv c_2 \pmod{n_2} \\ x \equiv c_3 \pmod{n_3}. \end{cases} \tag{1}$$

The CRT guarantees that $x = m^3 (\mathrm{mod}\, n_1 n_2 n_3)$. As $m^3 < n_1 n_2 n_3$, then $x = m^3$. It follows that the eavesdropper can recover m by computing the integer cube root of x.

Thus small encryption exponent such as $e = 3$ should not be used if the same message is sent to many entities. Alternatively, to prevent such an attack a pesudorandomly generated string should be appendend to the plaintext prior to encryption.

1.2 RSA with CRT

Let e, d, and n be, respectively, the encryption exponent, the decryption exponent, and the modulus. Let c a cryptogram obtained by encrypting message $m, m < n$ with e, i.e., $c = m^e \bmod n$. In order to recover m from c, the recipient has to compute $m = c^d \bmod n$. The same result can be obtained more efficiently using the CRT as follows.

Initially, the recipient computes m_1 and m_2 as follows

$$\begin{cases} m_1 = c^d \bmod p \\ m_2 = c^d \bmod q \end{cases} \tag{2}$$

[1] A passive adversary.

For the Little Fermat's Theorem, equation 2 can be re-written as follows

$$\begin{cases} m_1 &= c^{d \bmod p-1} \mod p \\ m_2 &= c^{d \bmod q-1} \mod q. \end{cases} \tag{3}$$

Finally, the recipient applies the CRT and computes

$$m = a_1 m_1 q + a_2 m_2 p \tag{4}$$

where $a_1 = q^{-1} \bmod p$ and $a_2 = p^{-1} \bmod q$. Notice that quantities a_1 and a_2 only depends on system parameters and thus can be precomputed.

The performance advantages are evident from the following observations. Let us suppose that the binary rapresentation of modulus n requires k bits. The exponent d is of the same order as n and thus its binary rapresentation requires k bits. It follows that computing $m = c^d \bmod n$ using the square-and-multiply algorithm takes time $O(k^3)$. In contrast exponents in equation 3 are of the same order as p and q, and thus their rapresentations are on $k/2$ bits. It follows that computing m using the CRT requires compute two exponentiations on $k/2$ bits and thus takes $O(\frac{k^3}{4})$. It follows that this approach is four times faster. Furthermore, it requires a smaller amount of memory for intermediate results, since modular exponentiation is performed on half the bit size.

1.2.1 Fault injection attack

Boneh, DeMillo, and Lipton present an elegant and simple theoretical attack on RSA with CRT implementations [BDL01]. With reference to equation 3, let us suppose that a fault is injected during the computation of m_1 so producing m_1' and thus m'. It follows that $m - m' = a_1 q(m_1 - m_1')$. Thus one can observe that $\gcd(m - m', n) = q$. It follows that q q can be efficiently computed by applying the Euclid's algorithm for greatest common divisor to n and $m - m'$.

This result comes as a surprise, considering that the RSA algorithm resisted years of pure mathematical cryptanalysis [1]. Indeed, the mathematical constructions do not take into account faulty computations, but assume that a computation always results in the correct output. A physical implementation, however, may perform faulty computations. This is a known case, especially for space missions, where the environment can be somewhat hostile. Bit flips can occur for many reasons, such as radiation, power supply or clock manipulation, or even incomplete testing of the hardware during production. While such events might be rare in nature, an attacker can actively force the device to operate in such an environment. Embedded systems can be more vulnerable to such attacks because their resource-constrained environment makes it easy for an attacker to inject a fault in the system. These vulnerabilities are worrisome for financial or commercial applications, such as smart cards for banking (credit cards), cellular phone SIMs, and pay-per-view TV.

Since the introduction of fault-injection attacks, several countermeasures have been proposed. The first proposal performs double computation. In order to detect faulty computations, this proposal computes the result twice before providing an output. This is not

always efficient because it splits latency and throughput for a given system. Furthermore, this approach cannot detect permanent faults in which a specific memory area is stuck to a value. The second proposal shows complementary operation of the algorithm. For example, signature verification ensures original input of the algorithm. Signature verification can be rather time-consuming in an embedded system because low value exponents are used for the operation performed by the system and a high value exponent is used by the verifying system.This verification results in higher execution time and complexity.

This field of research has been quite active in recent years. Most, if not all, of the proposed changes in the implementation of the RSA with CRT algorithm, however, were vulnerable to some form of side-channel attack. In many cases, the additional checks and countermeasures inserted in the implementation created additional side channels which can be more easily utilized by attackers instead of protecting the implementation of the algorithm. For a complete treatment of the subject, interested readers should consult.

References

[BDL01] D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 14(2):101–119, 2001.