6-2005

Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults

Feng BAO
Singapore Management University, fbao@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Y. HAN

A. JENG

Arcot Desai NARASIMHALU
Singapore Management University, desai@smu.edu.sg

See next page for additional authors

Citation

BAO, Feng; DENG, Robert H.; HAN, Y.; JENG, A.; NARASIMHALU, Arcot Desai; and NGAIR, T.. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. (2005). *Security Protocols: 5th International Workshop: Paris, France, April 7-9, 1997: Proceedings*. 1361, 115-124.
Available at: https://ink.library.smu.edu.sg/sis_research/3723

Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults

F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, T. Ngair

Institute of Systems Science
National University of Singapore
{baofeng, deng, yfhan, jeng, desai, teowhin}@iss.nus.sg

Abstract. In this paper we present a method of attacking public-key cryptosystems (PKCs) on tamper resistant devices. The attack makes use of transient faults and seems applicable to many types of PKCs. In particular, we show how to attack the RSA, the ElGamal signature scheme, the Schnorr signature scheme, and the DSA. We also present some possible methods to counter the attack.

1 Introduction

In September 1996, Boneh, DeMillo and Lipton from Bellcore announced a new type of cryptanalytic attack against RSA-like public key cryptosystems on tamper resistant devices such as smart card [4]. However, technical details of the Bellcore attack were withheld in that announcement and was released only at the end of October 1996. On 18th October 1996, Biham and Shamir published their attack, called Differential Fault Analysis (DFA), to secret key cryptosystems [5], such as DES. Some concrete ideas on how their attack works were revealed in their announcement.

Our work here was motivated first by the Bellcore announcement and then by the DFA announcement. Our first report on attacking RSA and some countermeasures were posted in the Internet on the 23rd and 24th October 1996 [2]. Right after that, A. K. Lenstra sent us his memo [9] on attacking RSA in Chinese remainder in a private communication. Subsequently, we released a more complete research note on attacking RSA and the ElGamal signature scheme on the 29th October 1996 [3]. Recently, Joye and Quisquater extended the Chinese remaindering attack to LUC and Demytko cryptosystems [8].

In this paper, we continue our earlier effort of attacking public-key cryptosystems (PKCs) on tamper resistant devices. Our attacking model makes use of the transient faults and seems applicable to many types of PKCs, such as RSA-like schemes and discrete logarithm based schemes. As in the Bellcore and DFA announcements, we assume that by exposing a sealed tamper resistant device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce with reasonable probability faults at random bit locations in a tamper resistant device at some random intermediate stage in the cryptographic computation. The faults in the random bit locations do not influence the code itself, i.e., the program itself does not crash, and only some of the values it operates upon are affected. It is further assumed that the attacker is in physical possession of the tamper resistant device and that he can repeat the experiment with the same private key by applying external physical effects to obtain faulty outputs.

The organization of the paper is as follows. In Section 2, we first report our attacks to RSA, and then present Lenstra's attack to RSA implemented based on the Chinese Remainder Algorithm (CRA). In Section 3, we show how to break discrete logarithm based schemes such as the ElGamal signature scheme [7], the Schnorr signature scheme [11], and the Digital Signature Algorithm (DSA). At the end of each section, we also give some possible methods to counter the attack.

When this manuscript was near its completion, Dan Boneh kindly sent us their paper [6] in a private communication. Throughout this paper, we will make remarks about the relation between their paper and our work wherever it is appropriate.

2 Attacking the RSA Scheme

Let $n = pq$ be the product of two primes p and q in RSA, e be the public exponent which is publicly known and d be the private exponent which is stored inside the tamper resistant device. Our attacks to RSA will be described in terms of ciphertext decryption although they can also be described in terms of signature generation.

Let m be a plaintext, then the corresponding ciphertext is

$$c \equiv m^e \bmod n$$

Denote the binary representation of the private exponent as $d_{t-1}|d_{t-2}|\cdots|d_i|\cdots|d_1|d_0$, where d_i, taking value 1 or 0, is the ith bit, t is the number of bits in d, and $x|y$ denotes concatenation of x and y. Further, we denote

$$c_i \equiv c^{2^i} \bmod n, \text{ for } i = 0, 1, 2, ..., t-1$$

Given c and d, the corresponding plaintext m can be expressed as

$$m \equiv c^d \bmod n \equiv c_{t-1}^{d_{t-1}} \cdots c_i^{d_i} \cdots c_1^{d_1} c_0^{d_0} \bmod n$$

2.1 Attack I

For the sake of simplicity, here we assume that in decrypting a ciphertext a single bit error is induced in c_i, for a random $i \in \{0, 1, 2, ..., t-1\}$. Denote the corrupted value as c_i'. Then the output from the tamper resistant device is

$$m' \equiv c_{t-1}^{d_{t-1}} \cdots c_i'^{d_i} \cdots c_1^{d_1} c_0^{d_0} \bmod n$$

The attacker now has both m and m' so that he is able to compute

$$\frac{m'}{m} \equiv \frac{c_i'^{d_i}}{c_i^{d_i}} \bmod n$$

which equals $\frac{c_i'}{c_i} \bmod n$ if $d_i = 1$ or equals 1 if $d_i = 0$. (From now on we assume that every number we meet is relatively prime with respect to n, hence we can compute its inverse.) The attacker can easily compute all the possible $\frac{c_i'}{c_i} \bmod n$ values in advance (there are a total of t^2 such values since c_i' has t possible values). Now the attacker compares all these values with $\frac{m'}{m} \bmod n$. Once a match is found, he knows i and then knows that d_i is 1.

This simple example is just meant to illustrate the basic ideas of our attack. It showed that one bit fault at certain location and time can cause fatal leakage of the private key.

The example above assumes that only one c_i contains a single bit error and that there is no error propagation from c_i to c_j, $j > i$. The effects of such error propagation were considered in [6] and [9]. As a result, the error models in [6] and [9] are more complicated and probably more realistic than ours. From practical viewpoint, our model can be explained as the model for "read" error. That is, c_i is mistaken as c_i' when it is multiplied to the value for computing c^d but remains correct when it is squared to obtain c_{i+1}.

Another issue is that we can actually consider multi-bit faults instead of one bit fault only. In this case, we need to compare $m'/m \bmod n$ with many more possible values. For the case of two-bit faults, $m'/m \bmod n$ should be matched with all the values $c_{i_1}'c_{i_2}'/c_{i_1}c_{i_2} \bmod n$ ($i_1, i_2 \in \{0, 1, 2, ..., t-1\}$) and $c_i''/c_i \bmod n$, where c_i'' denotes the value of two bit errors in c_i. In this case, $O(t^4)$ possible values should be generated in advance and matched (as well as those $c_i'/c_i \bmod n$) with the value $m'/m \bmod n$. In general, about t^{2j} values need to be generated in the situation where j-bit faults may take place.

2.2 Attack II

Suppose that one bit in the binary representation of d is flipped and that the faulty bit position is randomly located. An attacker arbitrarily chooses a plaintext m and computes the ciphertext c. He then asks the tamper resistant device to decrypt c and induces a random bit error in d by applying external physical effects to the device. Assuming that d_i is changed to its complement d_i', then the output of the device will be

$$m' \equiv c_{t-1}^{d_{t-1}} \cdots c_i^{d_i'} \cdots c_1^{d_1} c_0^{d_0} \bmod n$$

Since the attacker now possesses both m and m', he can compute

$$\frac{m'}{m} \equiv \frac{c_i^{d_i'}}{c_i^{d_i}} \bmod n.$$

Obviously, if $m'/m \equiv 1/c_i \bmod n$, then $d_i = 1$, and if $m'/m = c_i \bmod n$, then $d_i = 0$. Therefore, the attacker can compare $m'/m \bmod n$ to $c_i \bmod n$ and $c_i^{-1} \bmod n$, for $i = 0, 1, ..., t-1$, in order to determine one bit of d. He repeats the above process using either the same plaintext/ciphertext pair or using different plaintext/ciphertext pairs until enough information in d is obtained.

Suppose one bit error takes place randomly in d in each fault test. Then by basic probabilistic counting, we have the following: If we take $t \log t$ fault tests, with a probability larger than half, every bit of d is disclosed.

It should be noted again that this attack applies to the case of multiple bit errors. Assuming two bit faults. The attacker needs to compare $m'/m \bmod n$ with $c_i c_j \bmod n$, $c_i/c_j \bmod n$, and $1/(c_i c_j) \bmod n$, for all $i, j \in \{0, 1, 2, ..., t-1\}$. In this case, matching $m'/m \bmod n$ with all these values has a complexity of $O(t^2)$ instead of $O(t)$ as in the single error case; while with large possibility one obtain two bits, d_i and d_j, once a successful match is obtained.

2.3 Lenstra's Attack on RSA with Chinese Remainder Algorithm

Boneh, DeMillo and Lipton [6] gave an attack on RSA implemented with the CRA. Their attack requires two signatures of a given message: one correct signature and one faulty signature. Lenstra independently worked out a similar attack against RSA with CRA which requires only one faulty signature of a known message [9]. In the following, we briefly outline Lenstra's attack.

The signature s of a message m equals $m^d \bmod n$ and thus $s^e \bmod n$ is again equal to $m \bmod n$. It is well known that s can be computed by computing

$$u \equiv m^d \bmod p \text{ and } v \equiv m^d \bmod q,$$

and by combining u and v using the CRA. If a fault occurs in the course of the computation of the signature, the resulting value, denoted as s', will most likely not satisfy $m \equiv s'^e \bmod n$. If, however, the fault occurred only during the computation of say, u, and if v and the CRA were carried out correctly, then the resulting faulty signature s' satisfies $s'^e \equiv m \bmod q$, but the same congruence mod p does not hold. Therefore, q divides $s'^e - m$ but p does not divide $s'^e - m$, so that a factor of n may be discovered by the recipient of the faulty signature s' by computing the greatest common divisor of n and $s'^e - m$. This attack is very powerful since it requires only one faulty signature and it works under a general fault model.

2.4 Some Possible Countermeasures

There may be a variety of attacks to PKCs by inducing faults. The means of breaking a PKC can be devised to be dependent on the specific PKC algorithm as well as on its implementation. Generally speaking, countermeasures to such attacks are relatively insensitive to both the implementation of a PKC and the attacking scenarios. Here we envisage two general approaches to counter such attacks, one is based on the principle of "check and balance" and the other based on the principle of "information hiding". The former can be done by checking/verifying the result before sending it to the outside world and the latter can be achieved by introducing some randomness in the intermediate stages of the cryptographic computation.

a) The attacks may be avoided by calculating the output 2 times and matching the two results. However, this approach doubles the computational time. As pointed out in [4], this double computation method also avoids their attack. The weakness of this counter measure is that it slows down the computation by a factor of 2, which is "not accepted for some applications"[4].
b) In many cases, the encryption key e is usually small. So we can verify the result by checking $m'^e = c \bmod n$? It is much more efficient than the double computation approach if e is small. This approach was also pointed out independently by Lenstra.
c) In some protocols for digital signature, a random string is chosen by the smart card and concatenated to a message m which is to be signed by the smart card. For example, m is a 412 binary string given to the smart card. The smart card randomly chooses a 100 bit number r and the output is $(m|r)^d \bmod n$. Since r is different each time, the attack does not work in such case.
d) In the case where e is large and where the tamper resistant device is required to compute $c^d \bmod n$, the following efficient method may be used to counter the attack. The tamper resistant device generates a random number r and computes $r^d \bmod n$. This can be done in advance, i.e., before c is input and when the device is idle. To compute $c^d \bmod n$, the device first computes $rc \bmod n$, then $(rc)^d \bmod n$, and finally $\frac{(rc)^d}{r^d} \bmod n$. If no fault takes place, the output is obviously correct. If any fault takes place, the output is masked by r. Since r is unknown to the attacker and different for every decryption, our attack does not work. For the example, in the case of Attack II, if d_i is 0 and d'_i is 1, then $m'/m \equiv r^{2^i} c_i \bmod n$. Since r is unknown to the attacker, the ratio is useless to him.

It should be pointed out that a) - d) work against our attacks while only a) c) work against Lenstra's attack.

3 Attacking Discrete Logarithm Based Schemes

The general concept of attacking the RSA scheme can be applied to attack against discrete logarithm based public key cryptosystems. In the following, we show our attacks to the ElGamal signature scheme, the Schnorr signature scheme, and the DSA. Throughout this section, we will denote a signer's private key as x and its binary representation as $x_{t-1}|x_{t-2}|\cdots|x_i|\cdots x_1|x_0$, where t is the number of bits in x and x_i is the ith bit of x. The private key is kept inside a tamper resistant device and the corresponding public key can be made available to everyone.

The general steps followed by an attacker are as follows: 1) the attacker applies external physical effects to induce some bit errors at random locations in x and then obtains a faulty signature, 2) he performs some computations on the faulty signature to uncover part of the private key x. The attacker repeats steps 1) and 2) until he uncovers the binary representation of x or a sufficiently large number of bits that allow him to discover the rest of x by brute force. To keep the paper compact, we will only show steps 1) and 2) in the following, without explicitly showing the loops of the attack.

To simplify the description, we first show the attacks for the case of single bit error. We then briefly discuss the case of multiple bit errors.

3.1 Attacking the ElGamal Signature Scheme

In the ElGamal signature scheme [7], to generate a private and public key pair, we first choose a *prime* p, and two random numbers, g and x, such that both g and x are less than p. The private key is x and the public key is $(y \equiv g^x \bmod p, g, p)$.

To generate a signature on a message m, the signer first picks a random k such that k is relatively prime to $p-1$. She then computes

$$w \equiv g^k \bmod p \text{ and } s \equiv (m - xw)/k \bmod (p-1)$$

The signature is the pair w and s. To verify the signature, the verifier confirms that

$$y^w w^s \equiv g^m \bmod p.$$

Assume that x_i in x is changed to its complement x_i' during the process of signing of a message m. We denote the corrupted x as x' due to the flip of x_i. Then the outputs of the device will be

$$w \equiv g^k \bmod p \text{ and } s' \equiv (m - x'w)/k \bmod (p-1)$$

Using w, s', m, and the signer's public key (y, p, g), the attacker computes

$$T \equiv y^w w^{s'} \bmod p \equiv g^m g^{w(x-x')} \bmod p.$$

Let $R_i \equiv g^{w2^i} \bmod p$ for $i = 0, 1, 2, ..., t-1$. Then, we have

$$TR_i \equiv g^m \bmod p, \text{ if } x_i = 0$$

(since for $x_i = 0$ we have $x - x' = -2^i$) and

$$\frac{T}{R_i} \equiv g^m \bmod p, \text{ if } x_i = 1$$

(since for $x_i = 1$ we have $x - x' = 2^i$). The attacker computes TR_i and T/R_i and tests to see if either TR_i or T/R_i equals $g^m \bmod p$, for $i = 0, 1, ..., t-1$. If a match is found, then one bit of x is found.

3.2 Attacking the Schnorr Signature Scheme

In the Schnorr signature scheme [11], to generate a private and public key pair, we first choose two primes, p and q, such that $p = zq + 1$ for a reasonably large q. We then select a number g not equal to 1, such that $g^q \equiv 1 \bmod p$. The signer's private key is a random x less than q, and the public key is $(y \equiv g^{-x} \bmod p, g, p, q)$.

To generate a signature on a message m, the signer first picks a random k that is less than q. She then computes

$$w \equiv g^k \bmod p, e = h(m|w) \text{ and } s \equiv ex + k \bmod q,$$

where h is a secure one-way hash function that outputs a number less than q. The signature is the pair e and s. Because

$$(g^s y^e \bmod p) = w,$$

to verify the signature, the verifier confirms that

$$h(m|(g^s y^e \bmod p)) = e$$

During the computation of s, assuming that x_i in x is flipped to x_i' and denote the corrupted x as x'. Then the outputs of the device will be

$$e \equiv h(m|w) \bmod p \text{ and } s' \equiv ex' + k \bmod q$$

Using e, s', m, and the signer's public key (y, p, g, q), the attacker computes

$$T \equiv g^{s'} y^e \equiv wg^{e(x'-x)} \bmod p$$

Let $R_i \equiv g^{e2^i} \bmod p$ for $i = 0, 1, 2, ..., t-1$. It is easy to see that $TR_i \equiv wg^{e(x'-x+2^i)} \bmod p$ and $T/R_i = wg^{e(x'-x-2^i)} \bmod p$. Then we have

$$h(m|(TR_i \bmod p)) = e, \text{ if } x_1 = 1$$

(since for $x_i = 1$, we have $x' - x = -2^i$ and then $TR_i \equiv w \bmod p$), and

$$h(m|(T/R_i \bmod p)) = e, \text{if } x_i = 0$$

(since for $x_i = 0$, we have $x' - x = 2^i$ and then $T/R_i \equiv w \bmod p$). Therefore, by iterating through different i and matching e with $h(m|(TR_i \bmod p))$ and $h(m|(T/R_i \bmod p))$, the attacker can discover the ith bit, x_i, of the private key x.

In [6], Boneh, DeMillo and Lipton gave an attack against the Schnorr identification scheme [11]. In their attack, it was required that 1) the verifier uses the same challenge e (which plays the same role as the e in the Schnorr signature scheme) in all invocations of the identification protocol without being detected by the prover's tamper resistant device, and 2) a bit error is introduced in the random number k (which plays the same role as the k in the Schnorr signature scheme). Because of these two requirements, their attack can not be applied to break the Schnorr signature scheme. On the other hand, our attack to the Schnorr signature scheme can be applied to break the Schnorr identification scheme with little modification.

3.3 Attacking the DSA

In the DSA, to generate a private and public key pair, we first choose a prime p such that $p = zq + 1$ for a reasonably large prime q. We then compute $g \equiv b^{(p-1)/q} \bmod p$, where b is any number less than $p-1$ such that $(b^{(p-1)/q} \bmod p)$ is greater than 1. The signer's private key is x, a random number less than q, and the public key is $(y \equiv g^x \bmod p, g, p, q)$.

To sign a message m, the signer first picks a random k that it is less than q. She then computes

$$w \equiv g^k \bmod p \bmod q \text{ and } s \equiv (e + wx)/k \bmod q,$$

where $e = h(m)$ with h being a secure one-way hash function that outputs a number less than q. The signature is the pair w and s. To verify the signature, the verifier confirms that

$$w \equiv g^{(ue \bmod q)} y^{(uw \bmod q)} \bmod p \bmod q,$$

where $u \equiv 1/s \bmod q$.

The attacker applies external physical effects to the tamper resistant device and at the same time asks the device to sign a message m. During the process of calculating s, we assume that the ith bit of x is changed from x_i to its complement x_i'. Let x' denote the corrupted x due to the flip of x_i. Then the outputs of the device will be

$$w \equiv g^k \bmod p \bmod q \text{ and } s' \equiv (e + wx')/k \bmod q$$

Using w, $u' \equiv 1/s' \bmod q$, m, and the signer's public key (y, p, g, q), the attacker can compute $e = h(m)$ and

$$T \equiv g^{(u'e \bmod q)} y^{(u'w \bmod q)} \equiv g^{(u'(e+xw) \bmod q)} \bmod p \bmod q.$$

Let $R_i \equiv g^{(u'w2^i \bmod q)} \bmod p \bmod q$ for $i = 0, 1, 2, \ldots, t-1$. Then we have

$$TR_i \equiv g^{(u'(e+w(x+2^i)) \bmod q)} \bmod p \bmod q$$
$$T/R_i \equiv g^{(u'(e+w(x-2^i)) \bmod q)} \bmod p \bmod q.$$

It is easy to show that

$$TR_i \equiv w \bmod p \bmod q, \text{ if } x_i = 0 \qquad T/R_i \equiv w \bmod p \bmod q, \text{ if } x_i = 1$$

So by iterating through different i and matching w with $TR_i \bmod p \bmod q$ and $T/R_i \bmod p \bmod q$, the attacker can discover the value of x_i.

3.4 Multiple Bit Errors

Extension of the above methods in attacking discrete log based digital signature schemes to the cases of multiple faults in x is straightforward. In the case of single fault, the Rs, denoted as R_i in the above single bit error case, each has a single argument i. Their computation and subsequent comparison is of complexity $O(2t)$. In the case of $j > 1$ faults in x, the Rs, which we will denote as $R_{i1,i2,\ldots,ij}$, will each have j arguments and their computations and subsequent comparisons are of complexity $O((2t)^j)$.

3.5 Countermeasures

The countermeasure achieved through double computation as mentioned in section 2.4 also applies to the attacks presented in this section.

Another countermeasure to the attack against discrete log based signature schemes is that the tamper resistant device stores both x and $1/x$, where x is used in the computation of s and $1/x$ is used to check the correctness of s. As an example let's consider the Schnorr signature scheme. Right after computing $s' \equiv ex' + k \bmod q$, the device verifies the value of s' by comparing e with $(s' - k)(1/x) \bmod q$. If these two values are the same, the result is considered correct; otherwise, the device is reset.

In general, to prevent corrupted variables from being used in a calculation and subsequently causing breaking of a cryptosystem, we suggest that the variable and its inverse be stored somewhere before the calculation takes place. The variable is used for the calculation and its inverse can be used to verify the result of the calculation.

To illustrate the above concept, let's again consider the Schnorr signature scheme. Suppose we want to make sure that the correct values of both x and k are used in the calculation of $s = ex + k$. The tamper resistant device stores $x, 1/x, k, 1/k$ somewhere before the calculation starts. After computing $s' = ex' + k'$, the device checks to see if $e = k(s'(1/k) - 1)(1/x)$. The value s' is considered correct only if the equality holds.

4 Concluding Remarks

The attack to public-key cryptographic schemes on tamper resistant devices presented in this paper makes use of transient faults. Our attacking model is independent of the implementation of a specific cryptosystem and seems to be applicable to breaking large classes of public-key cryptosystems. In particular, we showed how to break the RSA, the ElGamal signature scheme, the Schnorr signature scheme, and the DSA.

This attack highlighted that hardware faults can cause fatal leakage of the private/secret key values and may eventually lead to breaking of a cryptosystem. Therefore, it is important to take fault tolerance into serious consideration in the design of cryptosystems and to strike a balance between low overhead and high robustness. As a first step, we have proposed some methods to counter our attack. It should be noted that there are many other ways of breaking tamper resistant devices in addition to the ones outlined in this paper. In general, design of fault tolerant tamper resistant devices is a very challenging problem. Readers interested in getting more information in this area are refereed to the excellent paper by Anderson and Kuhn [1].

5 Acknowledgments

The work reported in this paper was motivated by the Bellcore Press Release [4] and further motivated by Biham and Shamir's research announcement [5].

The fundamental concept of breaking cryptosystems in the presence of hardware faults released in [4] was the main driving force behind our research. The authors would also like to thank Dr. Arjen Lenstra for his valuable comments on an earlier version of our manuscript.

References

1. R. Anderson and M. Kuhn, "Tamper Resistance - A Cautionary Note", to appear in the Proceedings of the 2nd Workshop on Electronic Commerce, Oakland, CA., Nov. 18-20, 1996.
2. F. Bao, R. Deng, Y. Han, A. Jeng, D. Narasimhalu, and T. Ngair, "Another New Attack to RSA on Tamperproof Devices", 23rd October. 1996, http:// www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/961022.sgtamper.html; "A Method to Counter Another New Attack to RSA on Tamperproof Devices", 24th October. 1996, http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/ 961024.sgtampercounter.html.
3. F. Bao, R. Deng, Y. Han, A. Jeng, D. Narasimhalu, and T. Ngair, "New Attacks to Public Key Cryptosystems on Tamperproof Devices", 29th October. 1996,http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/news-items/.
4. Bellcore Press Release, "New Threat Model Breaks Crypto Codes", Sept. 1996, http://www.bellcore.com/PRESS/ADVSRY96/facts.html.
5. E. Biham and A. Shamir,"Research Announcement: A New Cryptanalytic Attack on DES", 18th October 1996, http://jya.com/dfa.htm.
6. D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Computations", Submitted to Eurocrypt 96.
7. T. ElGamal, "A Public-Key Cryptosystems and a Signature Scheme Based on Discrete Logarithms", IEEE Trans. Information Theory, Vol. IT-31, No. 4, 1985, pp. 469-472.
8. M. Joye and J.-J. Quisquater, "Attacks on systems using Chinese remaindering", Technical Report CG-1996/9 of UCL, http://www.dice.ucl.ac.be/crypto/.
9. A. K. Lenstra, "Memo on RSA Signature Generation in the Presence of Faults", Manuscript, Sept. 28, 1996. Available from Author at arjen.lenstra@citicorp.com.
10. R. L. Rivest, A. Shamir, and L. M. Adleman,"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, vol. 21, No. 2, Feb. 1978, pp. 120-126.
11. C. Schnorr, "Efficient Signature Generation by Smart Cards", J. Cryptology, Vol. 4, 1991, pp. 161-174.