Cryptanalysis in the Presence of Hardware Faults

(Draft; Please do not distribute.)

Dan Boneh
dabo@bellcore.com

Richard A. DeMillo
rad@bellcore.com

Richard J. Lipton*
rjl@bellcore.com

Math and Cryptography Research Group, Bellcore,
445 South Street, Morristown NJ 07960

Abstract

We present a method for breaking various cryptographic schemes by taking advantage of random hardware faults. These techniques are especially effective against tamper proof devices. We intend to run the device in an extreme physical environment, e.g. in the presence of harmful radiation. As a result certain hardware faults may occur. These faults cause the device to generate erroneous data. We intend to use this erroneous data to extract the secret information stored in the device. Currently, our techniques are effective against systems such as RSA, Rabin's signature scheme and the Fiat-Shamir identification scheme.

1 Introduction

Direct attacks on the famous RSA cryptosystem seem to require that one factor the modulus. Therefore, it is interesting to ask whether there are attacks that avoid this. The answer is yes: the first was the recent attack based on timing [2]. It was observed that a few bits could be obtained from the *time* that operations took. This would allow one to break the system without factoring.

We have a new type of attack that also avoids directly factoring the modulus. We essentially use the fact that from time to time the hardware that is doing the multiplication *may* introduce errors. Our attack is most effective against an allegedly *tamper proof* device such as a smart card. The device may be subjected to disrupting radiation which may cause hardware faults. In fact, there are many ways of deliberately causing hardware faults. For instance, for a short period of time the device may be given a higher clock rate than it is designed for. Similarly, it may be given higher/lower voltage than what it was built for or subjected to high temperatures. The faulted values produced by the device enable us to break the underlying cryptographic protocol.

We consider a fault model in which faults are transient. That is, the hardware fault only effects the current data, but not subsequent data. For instance, a bit stored in a register might spontaneously flip. Or a certain gate may spontaneously produce an incorrect value. Note, that the change is totally silent: the hardware and the system have no clue that the change has taken place. We assume that the probability of such faults is small so that only a small number of them occur throughout the computation. As was noted above such faults may be induced by physical means.

Our attack is effective against several cryptographic schemes such as the RSA system and Rabin signatures [4]. As expected, the attack itself depends on the exact implementation of each of these

*Also at Princeton University. Supported in part by NSF CCR–9304718.

schemes. In this note we concentrate on attacking a specific widely used implementation of RSA. We show that given *one* faulted version of an RSA encrypted message one can efficiently factor the RSA modulus with high probability. The same approach can be used to break Rabin's signature scheme as well. The idea of using hardware faults to extract secret information from tamper proof devices can be generalized to other implementations of the RSA system. For more general implementations some degree of sophistication is required to mount the attack. We give the full details in the final version of the paper.

In Section 4 we show how hardware faults can be used to break the Fiat-Shamir identification scheme [3]. Given a few faulted values an adversary can completely recover the private key of the party trying to authenticate itself. This is an example of an actual protocol suitable for smart cards which can be broken using our new attack.

The new attack described in the paper somewhat reduces our confidence in tamper proof devices. Up until now it was believed that secret information stored in a tamper proof device can not be discovered by an adversary. This assumption was based on the difficulty of opening the device and examining its circuitry. Our new attack enables one to extract the secret information from the device without physically opening it. The attack exposes a serious flaw in the assumptions made by smart card manufacturers.

2 Breaking Chinese remainder based implementations

2.1 The RSA system

The RSA system can be implemented in a tamper proof device and used to perform various functions for its owner, Alice. Usually the device contains a secret RSA decryption key and is used to decrypt various messages for Alice. There are many applications where such tamper proof devices are used. For instance, the device can be used to obtain digital signatures for Alice. It can be used to authenticate Alice to a third party. And it can be used to decrypt incoming encrypted messages.

In all the applications described above some secret information is embedded in the device. Since the device is tamper proof, i.e. it can not be opened and its contents examined, it is assumed that the secret information stored in the device can not be extracted. We show how for RSA and other systems, the secret information stored in the device can be extracted without opening the device.

To fix our terminology from here on we say that the device is used to obtain digital signatures for Alice. Let $N = pq$ be a product of two large prime integers. To sign a message x using RSA the device computes $x^s \pmod N$ where s is a secret exponent. Here the message x is assumed to be an integer in the range 1 to N (in general one first hashes the message to an integer in that range). The security of the system relies on the fact that factoring the modulus N is hard. In fact, if the factors of N are known then one can easily break the system, i.e. sign documents without prior knowledge of the secret exponent.

The computationally expensive part of signing using RSA is the modular exponentiation of the input x. For efficiency many implementations exponentiate as follows: using repeated squaring they first compute $E_1 = x^s \pmod p$ and $E_2 = x^s \pmod q$. They then use the Chinese remainder theorem to compute the signature $E = x^s \pmod N$. We explain this last step in more detail. Let a, b be two precomputed integers satisfying:

$$\begin{cases} a \equiv 1 \pmod p \\ a \equiv 0 \pmod q \end{cases} \quad \text{and} \quad \begin{cases} b \equiv 0 \pmod p \\ b \equiv 1 \pmod q \end{cases}$$

Such integers always exist and can be easily found given p and q. It now follows that

$$E = aE_1 + bE_2 \pmod{N}$$

Thus, the signature E is computed by forming a linear combination of E_1 and E_2. This exponentiation algorithm turns out to be more efficient than using repeated squaring modulo N since the numbers involved are smaller. In the next section we describe how one can attack this type of RSA implementation using hardware faults.

2.2 Breaking RSA using hardware faults

Our attack on a tamper proof device using the above implementation of the RSA system will enable us to factor the modulus N. Once the modulus is factored the system is considered to be broken. Our attack is based on obtaining two signatures of the same message. One signature is produced while the device is operating under normal conditions and is therefore the correct signature. The other is obtained while the device is placed in an extreme environment which is likely to cause hardware faults.

Let M be a message and let $E = M^s \pmod{N}$ be the correct signature of the message. Let $\hat{E}$ be a faulted signature. Recall that E and $\hat{E}$ are computed as

$$E = aE_1 + bE_2 \pmod{N} \quad \text{and} \quad \hat{E} = a\hat{E}_1 + b\hat{E}_2 \pmod{N}$$

Since hardware faults occur with low probability it is likely that a hardware fault occurs only during the computation of *one* of $\hat{E}_1, \hat{E}_2$. That is, we may assume that a hardware fault occurs during the computation of $\hat{E}_1$ but no fault occurs during the computation of $\hat{E}_2$, i.e. $\hat{E}_2 = E_2$. Observe that

$$E - \hat{E} = (aE_1 + bE_2) - (a\hat{E}_1 + b\hat{E}_2) = a(E_1 - \hat{e}_1)$$

Now, if $E_1 - \hat{E}_1$ is not divisible by p then

$$\gcd(E - \hat{E}, N) = \gcd(a(E_1 - \hat{E}_1), N) = q$$

and so N can be easily factored. Notice that if the factors of N are originally chosen at random then it is extremely unlikely that p divides $E_1 - \hat{E}_1$. After all, $E_1 - \hat{E}_1$ can have at most $\log N$ factors.

To summarize, we demonstrated above that using one faulted value and one correct value the modulus used in the RSA system can be efficiently factored. We note that the above attack works under a very general fault model. It makes no difference what type of fault or how many faults occur in the computation of E_1. All we rely on is the fact that faults occur in the computation modulo only one of the primes.

2.2.1 Rabin's signature scheme

Rabin's signature [4] is another cryptosystem which relies on the hardness of factoring for it's security. As before let $N = pq$ be a product of two large primes. To sign a document D one first hashes D to a number D' between 1 and N. The signer, who knows the secret factorization of the modulus N, then computes the square root of $D' \pmod{N}$. Thus, the signature is $S = \sqrt{D'} \pmod{N}$. Without the factorization of N, computing the modular square root of a number is as hard as factoring the modulus.

As before, the expensive part of signing using Rabin's scheme is the extraction of the modular square root. This is usually implemented using the Chinese remainder theorem in a similar fashion

to the one described above. To compute $S = \sqrt{D} \pmod{N}$ one first computes $S_1 = \sqrt{D} \pmod{p}$ and $S_2 = \sqrt{D} \pmod{q}$. This is done using standard square root algorithm modulo a prime. One then uses the Chinese remainder theorem to compute S. Using the same a and b defined above, S is computed as

$$S = aS_1 + bS_2 \pmod{N}$$

The exact same attack described in the previous section can be used to break a device using this implementation of Rabin signatures. We ask the device to sign the same message twice. One signature, S, is obtained under normal conditions and is therefore the correct one. The other signature, $\hat{S}$, is obtained in an extreme physical environment and is therefore likely to be faulty. The same analysis as before shows that $\gcd(S - \hat{S}, N)$ is likely to yield a factorization of N.

3 Register faults

From here on our attacks are based on a specific fault model which we call *register faults*. We view a tamper proof device as composed of some circuitry and a small amount of memory. The circuitry is responsible for performing the arithmetic operations. The memory (registers plus a small on chip RAM) is used to store temporary values.

Our fault model assumes that the circuitry contains no faults. On the other hand, a value stored in a register may be corrupted. With low probability, one (or a few) of the bits of the value stored in some register may flip. We will need this event to occur with sufficiently low probability so that there is some likelihood of the fault occurring exactly once throughout the computation. As before, all errors are transient and the hardware has no clue that the change has taken place.

One may wonder whether our fault model is a reasonable one. After all, under normal operating conditions hardware nowadays is virtually error free. However, when put in an extreme environment errors are likely to occur. The presence of certain radiation may not effect the circuitry at all while still causing certain register cells to flip spontaneously.

4 Attacking the Fiat-Shamir identification scheme

The Fiat-Shamir [3] identification scheme is an efficient method enabling one party, Alice, to authenticate it's identity to another party, Bob. They first agree on an n-bit modulus N which is a product of two large primes and a security parameter t. Alice's secret key is a set of invertible elements $s_1, \ldots, s_t \bmod N$. Her public key is the square of these numbers $v_1 = s_1^2, \ldots, v_t = s_t^2 \pmod{N}$. To authenticate herself to Bob they engage in the following protocol:

1. Alice picks a random r and sends $r^2 \bmod N$ to Bob.
2. Bob picks a random subset $S \subseteq \{1, \ldots, t\}$ and sends the subset to Alice.
3. Alice computes $y = r \cdot \prod_{i \in S} s_i \bmod N$ and sends y to Bob.
4. Bob verifies Alice's identity by checking that $y^2 = r^2 \cdot \prod_{i \in S} v_i \pmod{N}$.

We show that by using hardware faults one can recover Alice's secret keys $s_1, \ldots, s_t$. Here we view Alice as a tamper proof device and the secret keys are somehow embedded in the circuitry of the device. Bob, who is trying to discover the secret stored in the device, will execute the protocol several

times. Each time Bob observes the value $r^2 \bmod N$ generated by the device. Bob then picks a subset $S \subseteq \{1, \ldots, t\}$ and observes the value $y = r \cdot \prod_{i \in S} s_i$ returned from the device.

Now, suppose that due to a fault, one of the bits of the register holding the value r is flipped while the device is waiting for Bob to send it the set S. In this case, Bob receives the correct value $r^2 \pmod N$, however y is computed incorrectly by the device. Due to the fault, the device outputs:

$$\hat{y} = (r + E) \cdot \prod_{i \in S} s_i$$

where E is the value added to the register as a result of the fault. Observe that since Bob knows the value $\prod_{i \in S} v_i$ he can compute

$$(r + E)^2 = \frac{\hat{y}^2}{\prod_{i \in S} v_i} \pmod N$$

Since E is a binary number of low weight (i.e. a power of 2 or a sum of a few powers of 2), Bob can guess this value. If E is guessed correctly then Bob can recover r since

$$(r + E)^2 - r^2 = 2E \cdot r + E^2 \pmod N$$

and this linear equation in r can be easily solved. Bob's ability to discover the secret random value r is the main observation which enables him to break the system. Using the value of r and E Bob can compute:

$$\prod_{i \in S} s_i = \frac{\hat{y}}{r + E} \pmod N$$

To summarize, Bob can compute the value $\prod_{i \in S} s_i$ by guessing the fault value E and using the formula:

$$\prod_{i \in S} s_i = \frac{2E \cdot \hat{y}}{\frac{\hat{y}^2}{\prod_{i \in S} v_i} - r^2 + E^2} \pmod N$$

We now argue that Bob can verify that the fault value E was guessed correctly. Let T be the hypothesized value of $\prod_{i \in S} s_i$ obtained from the above formula. To verify this value Bob interacts with the device under normal conditions so that the device works properly. The point is that Bob uses the same set S which he used when the device generated the fault. The device will now correctly produce two values $(r')^2$ and $y' = r' \cdot \prod_{i \in S} s_i$. Now Bob simply checks that $(y')^2 = (r')^2 \cdot T^2$. We claim that with high probability there is only one low-weight value E for which the relation is satisfied. Therefore, if the relation is satisfied, Bob is very likely to have obtained the correct value of $\prod_{i \in S} s_i$.

Even in the unlikely event of two values E, E' satisfying the relation, Bob can still break the system. Observe that the relation $(y')^2 = (r')^2 \cdot T^2$ implies that $T^2 = \prod_{i \in S} s_i^2$. If there are two low-weight values E, E' generating two values $T, T', T \neq T'$ satisfying the relation then clearly $T^2 = (T')^2 \pmod N$. If $T \neq -T' \pmod N$ then Bob can already factor N. Suppose $T = -T' \pmod N$. Then since one of T or T' must equal $\prod_{i \in S} s_i$ (one of E, E' is the correct fault value) it follows that Bob now knows $\prod_{i \in S} s_i \pmod N$ up to sign. For our purposes this is good enough.

The testing method above enables Bob to check whether a certain value of E is the correct one. Since E is a low binary weight integer Bob can try all possible values for E. For instance, if we assume the fault is a single bit flip then there are only n possible values for E (since in this case $E = 2^k$ for some $1 \leq k \leq n$). By testing all possible values for E until the correct one is found Bob can compute $\prod_{i \in S} s_i$.

Observe that once Bob has a method for computing $\prod_{i \in S} s_i$ for various sets S of his choice, he can easily find $s_1, \ldots, s_t$. The simplest approach is for Bob to construct $\prod_{i \in S} s_i$ for singleton sets, i.e.

sets S containing a single element. If $S = \{k\}$ then $\prod_{i \in S} s_i = s_k$ and hence the s_i's are immediately found. However, it is possible that the tamper proof device might refuse to accept singleton sets S. In this case Bob can still find the s_i's as follows. We represent a set $S \subseteq \{1, \ldots, t\}$ by its characteristic vector $U \in \{0,1\}^t$, i.e. $U_i = 1$ if $i \in S$ and $U_i = 0$ otherwise. Bob picks sets $S_1, \ldots, S_t$ such that the corresponding set of characteristic vectors $U_1, \ldots, U_t$ form a $t \times t$ full rank matrix over Z_2. Bob then uses the method described above to construct the values $T_i = \prod_{i \in S_i} s_i$ for each of the sets $S_1, \ldots, S_t$. To determine s_1 Bob constructs elements $a_1, \ldots, a_t \in \{0,1\}$ such that

$$a_1 U_1 + \ldots + a_t U_t = (1, 0, 0, \ldots, 0) \pmod 2$$

These elements can be efficiently constructed since the vectors $U_1, \ldots, U_t$ are linearly independent over Z_2. When all computations are done over the integers we obtain that

$$a_1 U_1 + \ldots + a_t U_t = (2b_1 + 1, 2b_2, 2b_3, \ldots, 2b_t)$$

for some known integers $b_1, \ldots, b_t$. Bob can now compute s_1 using the formula

$$s_1 = \frac{T_1^{a_1} \cdots T_t^{a_t}}{v_1^{b_1} \cdots v_t^{b_t}} \pmod N$$

Recall that the values $v_i = s_i^2 \pmod N$ are publicly available. The values $s_2, \ldots, s_k$ can be constructed using the same procedure.

To summarize, we demonstrated that using t faulted values and $O(nt)$ arithmetic operations Bob can recover the secret values $s_1, \ldots, s_t$.

4.1 A modification of the Fiat-Shamir scheme

One may suspect that our attack on the Fiat-Shamir scheme is successful due to the fact that the scheme is based on squaring. Recall that Bob was able to compute the random value r chosen by the device since he was given r^2 and $(r + E)^2$ where E is the fault value. One may try to modify the scheme and use higher powers. We show that our techniques can be used to break this modified scheme as well.

The modified scheme uses some publicly known exponent e instead of squaring. As before, Alice's secret key is a set of invertible elements $s_1, \ldots, s_t \bmod N$. Her public key the set of numbers $v_1 = s_1^e, \ldots, v_t = s_t^e \bmod N$. To authenticate herself to Bob they engage in the following protocol:

1. Alice picks a random r and sends $r^e \bmod N$ to Bob.
2. Bob picks a random subset $S \subseteq \{1, \ldots, t\}$ and sends the subset to Alice.
3. Alice computes $y = r \cdot \prod_{i \in S} s_i \bmod N$ and sends y to Bob.
4. Bob verifies Alice's identity by checking that $y^e = r^e \cdot \prod_{i \in S} v_i \pmod N$.

When $e = 2$ this protocol reduces to the original Fiat-Shamir protocol. Using the methods described in the previous section Bob can obtain the values $L_1 = r^e \bmod N$ and $L_2 = (r + E)^e \bmod N$. As before we may assume that Bob guessed the value of E correctly. Given these two values Bob can recover r by observing that r is a common root of the two polynomials

$$x^e = L_1 \pmod N \quad \text{and} \quad (x + E)^e = L_2 \pmod N$$

Furthermore, r is very likely to be the only common root of the two polynomials. Consequently, when the exponent e is polynomial in n Bob can recover r by computing the GCD of the two polynomials. Once Bob has a method for computing r he can recover the secrets $s_1, \ldots, s_t$ as discussed in the previous section.

We note that the system can be broken even when a large exponent $e < N$ is used, by using a much larger collection of faults. We give the details in the final version of the paper.

5 Breaking other implementations of RSA

In Section 2.1 we described an efficient implementation of the RSA algorithm using the Chinese remainder theorem. We observed that this implementation can be easily broken in the presence of hardware faults. In this section we mention that using register faults it is possible to break other implementations of RSA as well. Let N be n-bit RSA composite and s a secret exponent. The exponentiation function $x \longrightarrow x^s \bmod N$ can be computed using either one of the following two algorithms:

- Algorithm I

  init $y \leftarrow x$; $z \leftarrow 1$.

  main For $k = 1, \ldots, n$.
  If k'th bit of s is 1 then $z \leftarrow z \cdot y \pmod N$.
  $y \leftarrow y^2 \pmod N$.

  Output z.

- Algorithm II

  init $z \leftarrow x$.

  main For $k = n$ down to 1.
  If k'th bit of s is 1 then $z \leftarrow z^2 \cdot x \pmod N$.
  Otherwise, $z \leftarrow z^2 \pmod N$.

  Output z.

For both algorithms we can prove that given $n \log n$ faulted values one can recover the secret exponent in polynomial time. More precisely, for a message $M \in Z_N$, given $n \log n$ faulted values output by a device computing the function $M^s \bmod N$ one can recover the secret exponent s in polynomial time in n. The device may employ either one of the exponentiation algorithms above. Here by faulted values we mean values obtained in the presence of register faults. We leave the proofs for the final version of the paper.

6 Protecting against an attack based on hardware faults

One can envision several methods of protection against the type of attack discussed in the paper. The simplest method is for the device to check the output of the computation before releasing it. For functions such as exponentiation, checking the output seems expensive as it is likely to require recomputation of the function. This factor of two slow down in performance may be unacceptable in some scenarios.

An alternative approach for protecting the computation is the use of blinding. The same technique protects against timing attack as well. To compute $x^s \bmod n$ the device first picks a random number r and computes $y = r^e \bmod n$ where e is the public exponent. The device then computes $(xy)^s/r \bmod n$. The result is $x^s \bmod n$ as required. When the public exponent e is small this blinding procedure may not be too expensive. Furthermore, it is currently unknown how to use hardware faults to break this scheme.

A more sophisticated defense involves making use of methods such as the Bellare-Rogaway signature scheme [1]. Recall that our attack is based on obtaining two signatures of the same message. Systems such as Bellare-Rogaway use random padding in a clever way to prevent us from ever obtaining two identical signatures of the same message.

7 Summary

We described a general attack on tamper proof devices implementing encryption schemes based on the Chinese remainder theorem. Specifically we showed how an implementation of the RSA system and Rabin signatures can be easily broken given faulty outputs. These results show that implementors basing their scheme on Chinese remainder methods should be wary.

The idea of using hardware faults to extract secret information from tamper proof devices can be generalized to other implementations of the RSA system as well as to other systems. For instance, we explained how the Fiat-Shamir identification protocol can be broken using hardware faults.

We note that for some schemes, such as Schnorr's identification scheme, we were not able to design an attack based on hardware faults. We say that a protocol is *fault resistant* if it can withstand this type of attack. Certainly schemes used in smart cards should be fault-resistant. Developing the necessary theory to prove that certain schemes are fault resistant is currently an open area of research.

References

[1] M. Bellare, P. Rogaway, "The exact security of digital signatures - How to sign with RSA and Rabin", in Proc. Eurocrypt 96.

[2] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", Proc. of Cyrpto 96.

[3] U. Feige, A. Fiat, A. Shamir, "Zero knowledge proofs of identity", Proc. of STOC 87.

[4] M. Rabin, "Digital signatures and public key functions as intractable as factorization", MIT Laboratory for computer science, Technical report MIT/LCS/TR-212, Jan. 1979.