Side Channel Attack Resistant Implementation of Multi-Power RSA using Hensel Lifting

Varad Kirtane, C.Pandu Rangan[*]
(varad.kirtane@gmail.com), (prangan55@gmail.com)

Abstract

Multi-Power RSA [1] is a fast variant of RSA [2] with a small decryption time, making it attractive for implementation on lightweight cryptographic devices such as smart cards. Hensel Lifting is a key component in the implementation of fast Multi-Power RSA Decryption. However, it is found that a naïve implementation of this algorithm is vulnerable to a host of side channel attacks, some of them powerful enough to entirely break the cryptosystem by providing a factorisation of the public modulus N. We propose here a secure (under reasonable assumptions) implementation of the Hensel Lifting algorithm. We then use this algorithm to obtain a secure implementation of Multi-Power RSA Decryption.

1 Introduction

The problem of Multi-Power RSA was posed in [1]. Multi-Power RSA is a variant of standard RSA [2] with the public modulus $N = p^{b-1}q$ as opposed to $N = pq$ in standard RSA. Multi-Power RSA is deemed a fast variant of RSA, as observed in [5], because it has lesser decryption time than standard RSA. Boneh et al.[5] found that a straightforward implementation of Multi-Power RSA Decryption could achieve a speedup by a factor of more than 2 over standard RSA. This algorithm is thus particularly suited for implementation in lightweight cryptographic devices such as smart cards where fast algorithms that use relatively few resources are needed. As smart cards are used for authentication, security of underlying algorithms is one of prime concerns. We shall see here that a naïve implementation of Multi-Power RSA is vulnerable to a host of side channel attacks. We shall also see that a secure implementation of Hensel Lifting is essential for a secure implementation of Multi-Power RSA Decryption. We will then provide a secure implementation of Hensel Lifting and use it to obtain a secure implementation of Multi-Power RSA Decryption. In the rest of the paper, we use the

[*]Both authors from Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai -600037, India.

following abbreviations for various side channel attacks: SPA: Simple Power Analysis, DPA: Differential Power Analysis, TA:Timing Attack, GA: Glitch Attack and FA: Fault Attack.

2 Multi-Power RSA

We reproduce here the essential algorithms for Multi-Power RSA from [5], with the standard terminology of p and q as the two primes, N as the public modulus, e as the encryption exponent and d as the decryption exponent.

Key Generation: The key generation algorithm takes as input a security parameter n and an additional parameter b. It generates an RSA public/private key pair as follows:

1. Generate two distinct $\lfloor n/b \rfloor$-bit primes, p and q, and compute $N \leftarrow p^{b-1}.q$.
2. Use the same public exponent e used in standard RSA public keys, namely $e = 65537$. Compute $d \leftarrow e^{-1}\ mod\ (p-1)(q-1)$.
3. Compute $r_1 \leftarrow d\ mod\ (p-1)\ and\ r_2 \leftarrow d\ mod\ (q-1)$. The public key is $< N, e >$; the private key is $< p, q, r_1, r_2 >$.

Encryption: Same as in standard RSA.

Decryption: To decrypt a ciphertext C using the private key $< p, q, r_1, r_2 >$ one does:

1. Compute $M_1 \leftarrow C^{r_1}\ mod\ p$ and $M_2 \leftarrow C^{r_2}\ mod\ q$. Thus ${M_1}^e = C\ mod\ p$ and ${M_2}^e = C\ mod\ q$.
2. Using Hensel lifting [12] construct an ${M_1}'$ such that $({M_1}')^e = C\ mod\ p^{b-1}$. Hensel lifting is much faster than a full exponentiation modulo p^{b-1}.
3. Using Chinese Remainder Theorem (CRT), compute an $M \in \mathbb{Z}_N$ such that $M = {M_1}'\ mod\ p^{b-1}$ and $M = M_2\ mod\ q$. Then $M = C^d\ mod\ N$ is a proper decryption of C.

Security in Theory: The security of Multi-Power RSA depends on the difficulty of factoring $N = p^{b-1}q$. Thus, one has to ensure that p as well as q do not fall within the capabilities of general factoring methods ([10][11][7]) or special ones (e.g. Elliptic Curve Method has an improvement for $N = p^2q$, see [8]). Usually, $N = p^2q$ with a 1024-bit N is used.

Security in Practice: Although we may ensure that the factors of N do not fall within the capabilities of any state-of-the-art factoring algorithm, a naïve implementation of Multi-Power RSA may be easily broken through side channel attacks. We refer the reader to [3], which traces the history of developement of a secure implementation of modular exponentiation and CRT-RSA decryption. Giraud[3] also provides side channel attacks for various implementations proposed in the literature and finally provides implementations of modular exponentiation and CRT-RSA decryption that are secure against Fault Attack (FA) and Simple Power Analysis (SPA). Kim et al.[9] further improve upon this and provide an implementation of CRT-RSA decryption secure against Glitch Attack (GA) as well.

3 First Version of Multi-Power RSA Decryption

From the theoretical discussion in the previous section, it is clear that a complete implementation of Multi-Power RSA Decryption requires the implementation of three separate but related algorithms, namely:

- Modular Exponentiation
- CRT-RSA Decryption.
- Hensel Lifting.

We now propose a version of Multi-Power RSA Decryption based on the countermeasures provided by [3] and [9] for modular exponentiation and CRT-RSA decryption. Only a straightforward implementation of Hensel Lifting is available in the literature, and we shall use the implementation provided by [6].

3.1 Hensel Lifting

We provide here a version of Hensel Lifting taken from [6], suitably modified for compatibility with the other two algorithms. Note that in the algorithm below, $C_p = C \ mod \ p^2$, $d_p = d \ mod \ (p-1)$ and $e_{inv_p} = e^{-1} \ mod \ p$.

- Name: HenselLift
- Input: C, d_p, p, x, e_{inv_p}, e.
- Output: (S'_{p^2}, S_{p^2}). // Note that $(S'_{p^2}, S_{p^2}) = (x * C^{d+1} \ mod \ p^2, x * C^d \ mod \ p^2)$
- Procedure:

```
1. $p^2 \leftarrow p * p$.
2. $S_{p^2} \leftarrow C_p^{d_p - 1} \ mod \ p$.
```

```
3. $K_0 \leftarrow S_{p^2} * C_p \ mod \ p$.
4. $A \leftarrow -K_0^e \ mod \ p^2$.
5. $A \leftarrow A + C \ mod \ p^2$.
6. $S_{p^2} \leftarrow S_{p^2} * A \ mod \ p^2$.
7. $S_{p^2} \leftarrow S_{p^2} * (e_{inv_p}) \ mod \ p^2$.
8. $S_{p^2} \leftarrow S_{p^2} + K_0 \ mod \ p^2$.
9. $S_{p^2} \leftarrow x * S_{p^2} mod \ p^2$.
10. $S'_{p^2} \leftarrow S_{p^2} * C \ mod \ p^2$.
11. return $(S'_{p^2}, S_{p^2})$.
```

We refer the reader to [6] for a proof of correctness.

3.2 FA-SPA-GA resistant modular exponentiation

The following version of modular exponentiation is by [9] (a slight modification of [3]).

- Name: FA-SPA-GA-Exp
- Input: C, d=$(d_{n-1}, \ldots, d_0)$ odd, N, x, k (32-bit random number).
- Output: $(C^{d+1} \ mod(k.N), C^d \ mod(k.N))$.
- Procedure:

```
1. $a_0 \leftarrow C, a_1 \leftarrow {a_0}^2 \ mod(k.N)$
2. for $i$ from $n-2$ to 1 do
     –   $a_{\overline{d_i}} \leftarrow a_{\overline{d_i}}.a_{d_i} \ mod(k.N)$
     –   $a_{d_i} \leftarrow {a_{d_i}}^2 \ mod(k.N)$
3. $a_1 \leftarrow x.a_1.a_0 \ mod(k.N)$
4. $a_0 \leftarrow {a_0}^2 \ mod(k.N)$
5. $a_0 \leftarrow a_1.C \ mod(k.N)$.
6. if ($i$ & $d$ not disturbed) then return $(a_0, a_1)$.
   else return ("A fault attack has been detected.")
```

Here x is a randomly chosen element of $\mathbb{Z}_{kN}{}^*$. We note that the original paper by [3], this factor was not multiplied. Its inclusion is suggested by [9], to counter GAs on CRT-RSA algorithm, when this algorithm is used as a subroutine in it. We refer the reader to [9] for an in-depth discussion of this countermeasure. Also note that in the algorithm we will have to use $k = 1$ for compatibility with the Hensel Lifting algorithm.

3.3 FA-GA resistant CRT-RSA

This algorithm also comes from [9] (a slight modification of [3]). Again, we impose the restriction that k be 1 for compatibility with Hensel Lifting algorithm. Notice that this prevents us from blinding the modulus N. Also note that we provide the algorithm for the case of $N = p^2q$ as opposed to $N = pq$ mentioned in [9]. In the algorithm below, $d_p = d\ mod\ p$, $d_q = d\ mod\ q$.

- Name: FA-GA-CRT-RSA
- Input: C, p^2, q, d_p, d_q, a, e_{inv_p}, e
- Output: $C^d\ mod\ N$.
- Procedure:

```
1. Pick a 32-bit random number $x$.
2. $y = x^{-1}\ mod\ N$
3. $(S'_{p^2}, S_{p^2}) \leftarrow$ HenselLift with $C$, $d_p$, $p$, $x$, $e_{inv_p}$, $e$ as inputs.
4. $(S'_q, S_q) \leftarrow$ FA-SPA-GA-Exp with $C$, $d_q$, $q$, $x$, 1 as inputs.
5. $S' \leftarrow CRT(S_{p^2}', S_q')$
6. $S \leftarrow CRT(S_{p^2}, S_q)$
7. $S'' \leftarrow C.S\ mod(p^2.q)$
8. if $(S'' = S')$ & ($p^2$, $q$, $a$ not modified) then return $y.S$.
   else return ("A fault attack has been detected.");
```

In the above algorithm,

$$\begin{aligned} CRT(S_{p^2}, S_q) &= (((S_q - S_{p^2})\ mod\ q).a\ mod\ q).p^2 + S_{p^2}\ mod(p^2.q) \\ and \quad a &= p^{-2}\ mod\ q. \end{aligned}$$

Note that this multiplication by y is to be used to counter GA. It is noted in [9] that in the absence of this countermeasure, an adversary could attack the algorithm with a FA combined with a GA. We refer the reader to [9] for further discussion of this countermeasure.

3.4 Vulnerabilities

The first version of Multi-Power which we have proposed is not secure. First, notice that for compatibility with Hensel Lifting we are unable to blind the RSA decryption modulus N. This exposes it to SPA-type attacks as described in [13]. Thus, blinding the RSA modulus is necessary and Hensel Lifting must be modified to include blinding. It is, however, not sufficient as the Hensel Lifting algorithm is still vulnerable to FA as described below.

Suppose we were to inject a fault during any of the steps prior to the last three steps of the Hensel Lifting algorithm. We would then have $S_{p^2} \not\equiv S\ mod\ p^2$. Notice that however, the coherency between S_{p^2} and S'_{p^2} will still be maintained, because S'_{p^2} is calculated from S_{p^2}. Due to this, the coherency check done in the last step of FA-GA-CRT-RSA would fail to detect this problem and thus produce a faulty output S_F. Also, notice that $S_q \equiv S\ mod\ q$ would still hold. Thus, the output S_F of such an execution would have the following property:

$$S_F \not\equiv S\ mod(p^2)\ but\ S_F \equiv S\ mod\ q.$$

Hence, $S_F - S$ would be a multiple of q but not of p. Therefore $gcd(S_F - S, N)$ would reveal q and the cryptosystem would be totally broken in a mere two executions, a correct one to obtain S and a faulty one to obtain S_F. Note that this attack is highly practical and easy, as we make no assumption whatsoever about the nature or timing of the fault, except that it occurs in the Hensel Lifting algorithm prior to the last three steps. Thus, the first version of Multi-Power RSA Decryption is insecure.

4 Second Version of Multi-Power RSA Decryption

Having seen the attacks on the first version, we realise that it is mandatory that the Hensel Lifting algorithm be modified to allow blinding as well as be made secure against FA. We provide such a version of Hensel Lifting. We also reproduce the CRT-RSA decryption algorithm from [9], this time *with* the blinding of the RSA modulus.

4.1 FA-SPA-GA Resistant Hensel Lifting

This version of Hensel Lifting is secure against FA and SPA. The notation used is similar to the naïve implementation of Hensel Lifting above. Additionally, note that x is a random 32-bit number and k is a random 32-bit prime.

- Name: FA-SPA-GA-HenselLift
- Input: C, d_p, p, k, x, e.
- Output: (S'_{p^2}, S_{p^2}). // Note that $(S'_{p^2}, S_{p^2}) = (x * C^{d+1}\ mod\ p^2, x * C^d\ mod\ p^2)$
- Procedure:

```
1. $p^2 \leftarrow p * p$.
2. $(k.p)^2 \leftarrow k * k * p^2$
```

```
3. $e_{inv} \leftarrow e^{-1}\ mod\ (k.p)$
4. $(K_0, S_{p^2}) \leftarrow$ FA-SPA-GA-Exp with $C_p$, $d_p - 2$, $p$, 1, $k$.
5. $S_{p^2} = S_{p^2} * C_p\ mod\ (k.p)$
6. $S'_{p^2} = K_0 * C\ mod\ (k.p)^2$
7. $K_0 = K_0 * C_p\ mod\ (k.p)$
8. $K_0^{-1} \leftarrow K_0^{-1}\ mod\ (k.p)^2$
9. $(A_1, A_0) \leftarrow$ FA-SPA-GA-Exp with $K_0$, $e$, $p^2$, 1, $k$.
10. $A_0 = -A_0 + C\ mod\ (k.p)^2$
11. $A_1 = -A_1 + K_0 * C\ mod\ (k.p)^2$
12. $S_{p^2} = S_{p^2} * A_0 * e_{inv}\ mod\ (k.p)^2$
13. $S'_{p^2} = S'_{p^2} * A_1 * e_{inv}\ mod\ (k.p)^2$
14. $S_{p^2} = S_{p^2} + K_0\ mod\ (k.p)^2$
15. $S'_{p^2} = S'_{p^2} * K_0^{-1}\ mod\ (k.p)^2$
16. $S'_{p^2} = S'_{p^2} + K_0 * C\ mod\ (k.p)^2$
17. $S_{p^2} = x * S_{p^2}\ mod\ (k.p)^2$.
18. $S'_{p^2} = x * S'_{p^2}\ mod\ (k.p)^2$.
19. if ($p$ & $d_p$ not modified) & $(K_0 * A_0 = A_1)$ then return $(S'_{p^2}, S_{p^2})$.
    else return ("A fault attack has been detected.");
```

In order to see that this algorithm is indeed correct (in the absence of fault injections), we prove the following lemma:

Lemma 1. *For the same values of inputs, the values returned by FA-SPA-GA-HenselLift (with $k = 1$) are the same as the values returned by HenselLift (section 3).*

Proof. Let the values returned by HenselLift be (E, F) and those returned by FA-SPA-GA-HenselLift be (G, H). We trace back the steps of the algorithms to get the outputs in terms of inputs and internal variables. Note that all values are in modulo p^2. From HenselLift, we have:

$$\begin{aligned}
&(E,F) = (x * S_{p^2}, x * S_{p^2} * C) \ldots (from\ steps\ 9\ and\ 10)\\
&= (x * [S_{p^2} * e_{inv_p} + K_0], x * [S_{p^2} * e_{inv_p} + K_0] * C) \ldots (from\ steps\ 7\ and\ 8)\\
&= (x*[[S_{p^2}*[A+C]]*e_{inv_p}+K_0], x*[[S_{p^2}*[A+C]]*e_{inv_p}+K_0]*C) \ldots (from\ steps\ 5\ and\ 6)\\
&= (x * [(C_p^{d_p-1}\ mod\ p) * [-(K_0)^e + C]] * e_{inv_p} + K_0],\\
&\qquad x * [(C_p^{d_p-1}\ mod\ p) * [-(K_0)^e + C]] * e_{inv_p} + K_0] * C)\\
&\qquad\qquad\qquad \ldots (from\ steps\ 2, 3\ and\ 4)
\end{aligned}$$

From FA-SPA-GA-HenselLift (with $k = 1$, thus the modulus here is also p^2), we have:

$$\begin{aligned}
(G, H) &= (x * S_{p^2}, x * [S'_{p^2} + K_0 * C]) \ldots (\textit{from steps } 16, 17 \textit{ and } 18) \\
&= (x * [S_{p^2} + K_0], x * [S'_{p^2} * K_0^{-1} + K_0 * C]) \ldots (\textit{from steps } 14 \textit{ and } 15) \\
&= (x * [S_{p^2} * (-A_0 + C) * e_{inv} + K_0], \\
&\qquad x * [S'_{p^2} * K_0^{-1} * (-A_1 + K_0 * C) * e_{inv} + K_0 * C]) \\
&\qquad \ldots (\textit{from steps } 10, 11, 12 \textit{ and } 13) \\
&= (x * [S_{p^2} * (-A_0 + C) * e_{inv} + K_0], \\
&\qquad x * [S'_{p^2} * K_0^{-1} * (-K_0 * A_0 + K_0 * C) * e_{inv} + K_0 * C]) \\
&\qquad \ldots (\textit{from step } 9) \\
&= (x * [S_{p^2} * (-(K_0)^e + C) * e_{inv} + K_0], \\
&\qquad x * [S'_{p^2} * [-(K_0)^e + C] * e_{inv} + K_0 * C]) \\
&\qquad \ldots (\textit{from step } 9 \textit{ and } K_0^{-1} * K_0 = 1) \\
&= (x * [(C_p^{d_p - 1} \textit{ mod } p) * [-([C_p^{d_p} \textit{ mod } p])^e + C] * e_{inv} + [C_p^{d_p} \textit{ mod } p]], \\
&\quad x * [(C_p^{d_p - 1} \textit{ mod } p) * C * [-([C_p^{d_p} \textit{ mod } p])^e + C] * e_{inv} + [C_p^{d_p} \textit{ mod } p]] * C) \\
&\qquad \ldots (\textit{from steps } 4, 5, 6 \textit{ and } 7)
\end{aligned}$$

We notice that in HenselLift, $K_0 = C_p^{d_p} \textit{ mod } p$ and that $e_{inv_p} = e_{inv}$ when $k = 1$. We see that substituting $K_0 = C_p^{d_p} \textit{ mod } p$ in the equation for (E, F), we get exactly identical terms for (E, F) and (G, H). Hence, for the same values of inputs, the values returned by FA-SPA-GA-HenselLift (with $k = 1$) are the same as the values returned by HenselLift. $\square$

The above lemma shows the equivalence of the two algorithms for $k = 1$. It also shows that the return values of FA-SPA-GA-HenselLift have exactly the same form as HenselLift, except that calculations are done in modulo $k.p$ and modulo $(k.p)^2$ instead of modulo p and modulo p^2. Hence, for a k such that $gcd(e, k.p) = gcd(K_0, k.p) = 1$, the proof for a general k will be exactly like in [6], with calculations done in modulo $k.p$ and modulo $(k.p)^2$ instead of modulo p and modulo p^2. Thus, FA-SPA-GA-HenselLift is correct.

For the algorithm to remain practically implementable, we have to show that there is a very high probability that we can find a k that is relatively prime to e as well as K_0 in a few trials. We do this in the lemma below.

Lemma 2. *If the number of distinct prime factors of K_0 and e put together is m, then the number of trials t required to reduce the probability of failure to 2^{-c} (where c is a constant) is given by $t \geq \frac{c}{b - log_2(m.b)}$ where b is the number of bits of the random prime to be generated.*

Proof. We have that number of distinct prime factors $m = 2^{log_2(m)}$. Suppose we generate a b-bit prime number. Now, the approxiamte number of b-bit primes is given by

$$\frac{2^b}{b} = 2^{b-log_2(b)} \ (due\ to\ the\ prime\ number\ theorem).$$

Therefore,

$$Prob(failure\ after\ 1\ attempt) = \frac{2^{log_2(m)}}{2^{b-log_2(b)}} = 2^{log_2(m.b)-b}.$$

$$Hence, \qquad Prob(failure\ after\ t\ attempts) = 2^{t(log_2(m.b)-b)}.$$

We want this quantity to be less than or equal to 2^{-c}, giving:

$$c \leq t.(log_2(m.b) - b).$$

$$Hence, \qquad t \geq \frac{c}{b - log_2(m.b)}.$$

□

If we take reasonable values for various factors involved, say $c = 100, b = 32$ and $m = 128$, we get $t \geq 5$. Thus, after a mere 5 attempts we will be able to generate a suitable k with a probability of $1 - 2^{-100}$. It is also easy to see that due the logarithmic dependence on m, the function grows slowly if $m << 2^b$, which is true in any practical application.

4.2 FA-SPA-GA resistant CRT-RSA

This algorithm is very similar to the CRT-RSA algorithm mentioned previously. The only change made to this algorithm is introduction of blinding. The notation is the same as the previous version.

- Name: FA-SPA-GA-CRT-RSA
- Input: C, p, q, d_p, d_q, a, e.
- Output: $C^d\ mod\ N$.
- Procedure:

  1. Pick a 32-bit random number x and 32-bit random prime k.
  2. $y = x^{-1}\ mod(k.N)$
  3. $(S'_{p^2},S_{p^2}) \leftarrow$ FA-SPA-GA-HenselLift with C, d_p, p, k, x, e as inputs.
  4. $(S'_q,S_q) \leftarrow$ FA-SPA-GA-Exp with C, d_q, q, x, k as inputs.

```
5. $S' \leftarrow CRT_{blinded}({S_{p^2}}', {S_q}')$
6. $S \leftarrow CRT_{blinded}(S_{p^2}, S_q)$
7. $S'' \leftarrow C.S\ mod(p^2.q)$
8. if $(S'' = S')$ & ($p$, $q$, $a$ not modified) then return $y.S$.
   else return ("A fault attack has been detected.");
```

In the above algorithm,

$$\begin{aligned} CRT_{blinded}(S_{p^2}, S_q) &= (((S_q - S_{p^2})\ mod(k.q)).a\ mod(k.q)).p^2 + S_{p^2}\ mod(p^2.q) \\ and \quad a &= p^{-2}\ mod\ q. \end{aligned}$$

4.3 Security Argument

The FA-SPA-HenselLift algorithm we propose is secure against SPA as well as FA. Security from SPA comes due to the fact that the modulus is blinded by a random prime, preventing the attacks proposed by [13]. Note that the blinding is still effective even when we insist that k be a 32-bit random prime rather than a 32-bit random number, because there are abundant 32-bit primes (about 2^{27}, by the prime number theorem). This still keeps the number of possibilities sufficiently large.

To demonstrate security from FA, we recall that FA consists of introducing a fault in the internal variables of an algorithm. We will show that no matter which variable is disturbed, the disturbance will be caught by a coherence test. Any fault induced in K_0, A_0 or A_1 will cause the $(K_0 * A_0 = A_1)$ test to fail. Any fault induced in S'_{p^2} or S_{p^2} will disturb the coherence between them and cause the $(S'' = S')$ test in FA-SPA-GA-CRT-RSA to fail. Notice that because S'_{p^2} and S_{p^2} are calculated independently after the exponentiation stage, there is no probable way to disturb them and still maintain coherence between them. FA against public parameters will not work either, as the test which checks if they are modified will fail.

Security against GA comes due to multiplication by x (a random element) at the end. The argument is the same as in [9]. The security of FA-SPA-GA-exp and FA-SPA-GA-CRT-RSA itself comes from [3] and [9]. Regarding the security of the three algorithms when implemented together, we see from the above argument that any attack (GA,SPA,FA) is resisted by the algorithm itself or it causes the coherence test in FA-SPA-GA-CRT-RSA algorithm to fail.

4.4 Vulnerabilities

Although the above proposed algorithms are secure against FA, SPA and GA, they are not secure against DPA and TA. The attacker may run the above algorithms many times with a variety of inputs and make power as well as timing measurements. These measurements may help him learn secret information. This is explained in more detail in the next version of the Multi-Power RSA Decryption.

5 Final Version of Multi-Power RSA Decryption

Before discussing the final version, we make certain observations and comments on side channel attacks in general. These motivate the countermeasure added in the final version.

Major side channel attacks may be classified based on the level of abstraction at which countermeasures have to be proposed in order to effectively counter them. We consider two levels of abstraction, logic gate level and algorithmic level. Some measures are implemented at algorithmic level (FA, SPA, GA) whereas others require handling at both levels (TA, DPA). Note that GA may be dealt with purely at the algorithmic level [9] or purely at the logic gate level [4]. We mention that measures at the algorithmic level may require architectural support for implementation of the necessary algorithm(s).

Countering TA and DPA requires two-fold security. First, the control flow in the algorithms should be independent of data i.e. the algorithms should have only a single execution path. Second, the logic gates used for hardware implementation should have data-independent characteristics (leakage current, input-output delay, power consumption etc.). If the first property is absent, then the time taken or power consumed by different paths in the algorithm may be different. Thus, the attacker may execute the algorithms with different sets of inputs and figure out with path is followed for which data. Usually, the path taken depends on secret bits and thus these bits may be revealed. If the second property is absent, different inputs will have different power traces (or execution times). Again, the power consumed or time taken depends on secret bits in many cases and hence they may be compromised. We note that a large section of the literature on side channel attacks deals with obtaining and analysing power traces and times of execution.

We mention that the algorithms proposed in the second version have the first property of control flow being data-independent, as none of them have any branches during execution. Conditional statements appear only before the final answer is output. Thus, for a version secure against all major side channel attacks (FA, TA, SPA, DPA and GA) we only need logic gates that satisfy the second property above (i.e. having data-independent characteristics). Recently, a dynamic and differential logic style has been proposed in [4], that has signal independent switching behaviour. In [4], the authors show that during each clock cycle, power consumption and all circuit characteristics, such as leakage current, instantaneous current and input-output delay are identical and independent of the logic value and sequence of input data. We state that the secure Hensel Lifting algorithm we propose (as indeed any algorithm in this paper) must be implemented using the set of logic gates built by [4] for resistance against TA and DPA. Thus, the final version of Multi-Power RSA Decryption consists of algorithms specified in the second version implemented with the logic gates built by [4].

6 Possible Attacks

As explained in the security argument above, the implementations we provide are secure against all major side channel attacks. However, there exists an attack model in which our implementation is not secure, namely the *random transient fault attack* model. This fault model assumes that fault can occur at any point in the algorithm for a short time, and that the fault gets corrected after this duration. We believe that this model is too strong for an attacker to implement in practice. Also, the probability that this occurs by itself in the underlying hardware in a manner that is exploitable is negligible. Further, it is unclear whether or not such an attack is possible on the kind of logic gates proposed by [4]. In any case, there does not exist any implementation of any algorithm that is secure in this model.

7 Conclusion and Future Work

We looked at various implementations of algorithms required for Multi-power RSA Decryption available in the literature and found that they are vulnerable to side channel attacks. We proposed a secure implementation of Hensel Lifting and adapted the existing implementations of modular exponentiation and CRT-RSA decryption algorithms to make them secure against major side channel attacks as well as compatible with each other. The only attack model in which our implementation may be attacked is the *random transient fault attack* model. We leave open the problem of implementing Multi-Power RSA Decryption secure in this model.

References

[1] T. Takagi, *Fast RSA-type Cryptosystem Modulo $p^k q$*, Proceedings of Crypto 98 (ed. H. Krawczyk), vol. 1462 of LNCS, pp. 318-326. Springer-Verlag, 1998.

[2] R. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.* Communications of the ACM, 21(2):120-126. Feb. 1978.

[3] C. Giraud, *An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis*, IEEE Transactions on Computers, vol.55, no.9, pp.1116-1120, 2006.

[4] K. Tiri, M. Akmal, and I. Verbauwhede, *A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*, 29 European Solid-State Circuits Conference (ESSCIRC 2002), 2002.

[5] D. Boneh and H. Shacham, *Fast Variants of RSA*, RSA Laboratories Cryptobytes, Volume 5 No. 1, pp. 1-8, Winter/Spring 2002.

[6] C. Vuillaume, *Efficiency Comparison of Several RSA Variants*, url:citeseer.ist.psu.edu/591581.html, 2003.

[7] D. Boneh, G. Durfee and N. Howgrave-Graham, *Factoring $N = p^r q$ for Large r*, Proceedings of Crypto '99, vol. 1666 of LNCS, pp. 326-337, Springer-Verlag, 1999.

[8] E.Okamoto, R.Peralta, *Faster Factoring of Integers of a Special Form*, IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, E79-A, n.4 (1996).

[9] C.H. Kim and J.J. Quisquater, *Fault Attacks for CRT Based RSA: New Attacks, New Results and New Countermeasures*, Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems, Proceedings of First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007, vol.4462 of LNCS, pp. 215-228, Springer-Verlag, 2007.

[10] H.W.Lenstra, *Factoring integers with Elliptic curves*, Annals of mathematics 126, pp. 649-673, 1987.

[11] C.Pomerance, *The quadratic sieve factoring algorithm*, Advances in Cryptology - Proceedings of EUROCRYPT '84 vol.209 of Lecture Notes in Computer Science, Springer-Verlag, pp. 169-182, 1985.

[12] H.Cohen, *A Course in Computational Algebraic Number Theory*, vol. 138 of Graduate texts in Mathematics, Springer-Verlag, 1996.

[13] R. Novak, *SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation*, Proceeedings of Public Key Cryptography (PKC 2002), eds. D. Naccache and P. Paillier, pp. 252-262, 2002.

[14] P.Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems*, Advances in Cryptology, Proceedings of CRYPTO '96, ed. Neal Koblitz, pp. 104-113, 1996.