On the Implementation of a Fast Prime Generation Algorithm

Christophe Clavier and Jean-Sébastien Coron

[1] Gemalto, Security Labs,
La Vigie, Avenue du Jujubier, ZI Athélia IV,
F-13705 La Ciotat Cedex, France
christophe.clavier@gemalto.com
[2] University of Luxembourg,
Faculty of Sciences, Technology and Communication,
6, rue Richard Coudenhove-Kalergi,
L-1359 Luxembourg
jean-sebastien.coron@uni.lu

Abstract. A side-channel analysis of a cryptographic algorithm generally concentrates on the encryption or decryption phases, rarely on the key generation phase. In this paper, we show that, when not properly implemented, the fast prime generation algorithm proposed by Joye and Paillier at CHES 2006 is susceptible to side-channel analysis; its main application is the generation of RSA key-pairs for embedded platforms like smart-cards. Our attack assumes that some parity bit can be recovered through SPA when it appears in a branch condition. Our attack can be combined with Coppersmith's theorem to improve its efficiency; we show that for 1024-bit RSA moduli, one can recover the factorization of roughly 1/1000 of the RSA moduli.

Key-words : Simple Power Analysis, Prime generation algorithm, Coppersmith's theorem.

1 Introduction

Side-channel analysis, such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [7], generally concentrates on the encryption or decryption phases, rarely on the key generation phase. Namely encryption or decryption offer more flexibility to the attacker who can provide various messages as input and each time record a side channel leakage. In contrast, a key generation algorithm doesn't take any input (beyond a security parameter) and in general it doesn't help to execute it multiple times since a different key is obtained for each new execution.

In this paper, we show that, when not properly implemented, one of the fast prime generation algorithms proposed by Joye and Paillier at CHES 2006 [6] is susceptible to side-channel analysis. The main application of the Joye-Paillier algorithm is to generate RSA keys on embedded platforms like smart-cards, where efficiency is of crucial importance. Prime generation usually works by applying a primality test on randomly generated integers, until a prime is found. The technique described in [6] consists in generating random integers that are not divisible by the first primes p_i; then a prime appears with higher probability, and on average fewer primality tests have to be applied, which improves efficiency.

A faster variant is described in [6] where a sequence of candidates is generated from a random seed and the parity of each candidate is tested before applying a primality test, until a prime is found. In this paper, we concentrate on an implementation of this faster variant; we show that if n primality tests have been applied and if the parity bits can be obtained through SPA, then we can recover the $n-1$ least significant bits of the output prime. Coppersmith's theorem [1] shows that an RSA modulus $N = pq$ can be factored in polynomial time given half of the least significant (or most significant) bits of p. Therefore, if the number n of primality tests for p or q is more than half the bit-size of p, one can recover the factorization of N efficiently. We provide an analysis which shows that for certain parameters ($b_{min} = b_{max} = 0$) and for 1024-bit RSA moduli, this happens for 10^{-3} of the generated RSA moduli.

2 The Joye-Paillier Prime Generation Algorithm

The Joye-Paillier algorithm [6] consists in generating a sequence of candidates q that are co-prime with the first small primes p_i; a primality test $T(q)$ is then applied on each candidate until a prime is found. Here we concentrate on the faster variant (from Fig. 3 in [6]). One defines Π as the product of the r first primes, excluding $p_1 = 2$, so that Π is odd :

$$\Pi = \prod_{i=2}^{r} p_i$$

Let $[q_{min}, q_{max}]$ be the interval in which the prime integers must be generated. One defines integers b_{min}, b_{max} and v such that :

$$q_{min} \leq (v + b_{min})\Pi, \text{ and } \ (v + b_{max} + 1)\Pi < q_{max}$$

The prime integers will actually be generated in the sub-interval :

$$[(v + b_{min})\Pi, (v + b_{max} + 1)\Pi]$$

See [6] for more details on the selection of parameters b_{min}, b_{max} and v. We denote by $T(q)$ a primality test (for example, Miller-Rabin [8]).

Fast Prime Generation Algorithm [6] :
Parameters : Π odd, b_{min}, b_{max}, v
Output : a random prime $q \in [q_{min}, q_{max}]$

1. Compute $\ell \leftarrow v\Pi$
2. Randomly choose $k \in (\mathbb{Z}/\Pi\mathbb{Z})^*$

3. Randomly choose $b \in \{b_{min}, \ldots, b_{max}\}$ and set $t \leftarrow b\Pi$
4. Set $q \leftarrow k + t + \ell$
5. If (q `even`) then $q \leftarrow \Pi - k + t + \ell$
6. If $T(q) =$ `false` then
   (a) Set $k \leftarrow 2k \mod \Pi$
   (b) Go to step 4.
7. Output q

It is easy to see that the candidate q at step 6 is odd and co-prime with all primes in Π. Namely, we have that $q = \pm k \mod \Pi$ and k remains always in $(\mathbb{Z}/\Pi\mathbb{Z})^*$, because $2 \in (\mathbb{Z}/\Pi\mathbb{Z})^*$. This implies that q is co-prime with all primes in Π. Moreover, if $q = k+t+\ell$ is even at step 4, then $q' = \Pi-k+t+\ell = \Pi-2k+q$ must be odd since Π is odd, and therefore q is odd at step 6. Since we ensure that each candidate q is not divisible by the first small primes, each candidate is prime with higher probability, so one gets a faster prime generation algorithm (see [6] for a complete analysis).

3 Our Side-Channel Attack

Our attack is based on the assumption we can recover the parity of k at step 4 thanks to the parity test performed on q at step 5. Namely, on a practical implementation, a branch condition usually produces a different physical leakage depending on the result of the test. Indeed by measuring power consumption, the attacker may be able to determine if the operation $q \leftarrow \Pi - k + t + \ell$ has been performed or not, which gives him the parity bit of q. Our attack is therefore a Simple Power Attack (SPA) on the parity bit of q. In practice, this assumption may be realistic or not, depending on the micro-processor used, the presence of hardware countermeasures, and the way the test is implemented. We note that SPA attacks based on analogous assumptions have been described in [3, 4].

Here we show that this sequence of parity bits enables us to recover the least significant bits of the prime q returned as output. Our attack is based on the following simple lemma :

Lemma 1. *Let Π be an odd integer and let $k_0 \in \mathbb{Z}_\Pi$. Define $B_i = \lfloor 2^i\, k_0/\Pi \rfloor$ and the sequence $k_i = 2^i\, k_0 \bmod \Pi$. Then, for $i \geq 1$, $B_i = \sum_{j=1}^{i}(k_j \bmod 2)2^{i-j}$.*

Proof. By definition, we have $2^{i-1}\, k_0 = B_{i-1}\, \Pi + k_{i-1}$. Hence, we have $2^i\, k_0 = 2B_{i-1}\, \Pi + 2k_{i-1} = (2B_{i-1} + \lfloor 2k_{i-1}/\Pi \rfloor)\Pi + k_i$, which gives $B_i = 2B_{i-1} + \lfloor 2k_{i-1}/\Pi \rfloor$. It follows that $B_i = \sum_{j=1}^{i} \lfloor 2k_{j-1}/\Pi \rfloor 2^{i-j}$ (note that $B_0 = 0$).

Moreover, we have $2k_{j-1} = \lfloor 2k_{j-1}/\Pi \rfloor \Pi + k_j$. Taking the relation modulo 2, we get $\lfloor 2k_{j-1}/\Pi \rfloor \equiv k_j \pmod 2$ as Π is odd, and since $2k_{j-1} < 2\Pi$, we have $\lfloor 2k_{j-1}/\Pi \rfloor = k_j \bmod 2$. This concludes the proof. $\square$

Proposition 1. *With the previous notation and letting $b_j = k_j \bmod 2$, we have $k_i \equiv -(\sum_{j=1}^{i} b_j\, 2^{i-j})\, \Pi \pmod{2^i}$.*

Proof. This follows immediately from the previous lemma by observing that $B_i\, \Pi = 2^i\, k_0 - k_i$. □

We assume that the parameters Π, b_{min}, b_{max} and v are public and known to the attacker. Let $k_i = k_0 \cdot 2^i \mod \Pi$ for $i \geq 0$ denote the sequence of integers k which appear at step 4, where k_0 is the integer initially generated at step 2. Let $q_i = k_i + t + \ell$ denote the corresponding integer computed at step 4. From the parity of q_i tested at step 5, one can therefore obtain the parity of k_i; this is done by making an assumption on the parity of $t + \ell$, which is constant after step 4. Then using the previous lemma, after $n + 1$ primality tests the value of $k_n \mod 2^n$ is obtained [3]. Then we can write :

$$k_n = 2^n \cdot x + x_0$$

where $0 \leq x_0 < 2^n$ is known and x is unknown. Therefore the prime generated $q = k_n + t + \ell$ or $q = \Pi - k_n + t + \ell$ can be written :

$$q = 2^n \cdot x + x_0 + b \cdot \Pi + \ell$$

where x and $b \in [b_{min}, b_{max}]$ are integers unknown to the attacker, and the integers n, k_0, Π and ℓ are known.

We make the following assumption : we assume that $b_{min} = b_{max} = 0$, so that $b = 0$. We note that taking $b_{min} = b_{max} = 0$ is a valid parameter choice in the Joye-Paillier algorithm. In this case, the integer q can be written :

$$q = 2^n \cdot x + C$$

where C is a known constant and x is an unknown integer.

Theorem 1 (Coppersmith [1]). *Given $N = pq$ and the high-order or low-order $1/4 \log_2 N$ bits of p, one can recover the factorization of N in time polynomial in $\log N$.*

Using Coppersmith's theorem, one can therefore recover the factorization of N in polynomial-time if the number n of primality tests is at least half the bit-size of p. Let α denote the probability that a random n_0-bit odd integer q co-prime with Π is prime. We make the heuristic approximation that the candidates q_i behave as if they were uniformly and independently distributed. From the analysis of [5], we obtain that a candidate q_i is prime with probability :

$$\alpha = \frac{1}{n_0 \cdot \ln 2} \cdot \frac{\Pi'}{\phi(\Pi')}$$

where $\Pi' = 2\Pi$ and ϕ is Euler's function. Therefore, letting X be the random variable that gives the number of primality tests, we have :

$$\Pr[X = i] = (1 - \alpha)^{i-1} \cdot \alpha$$

[3] We need $n + 1$ parity bits because we don't use the first one.

which gives $E[X] = 1/\alpha$ and :

$$\Pr[X \geq i] = (1-\alpha)^{i-1}$$

To summarize, if the attack is applied for both p and q, the factorization of a $2n_0$-bit RSA modulus can be recovered in polynomial-time for a fraction :

$$\delta \simeq 2 \cdot \left(1 - \frac{1}{n_0 \cdot \ln 2} \cdot \frac{\Pi'}{\phi(\Pi')}\right)^{n_0/2-1}$$

of the generated moduli. We provide in Table 1 the corresponding fraction for various moduli size. For a prime size of n_0 bits, we take the largest r such that :

$$\Pi = \prod_{i=2}^{r} p_i < 2^{n_0-1}$$

Then we take $v = \lfloor 2^{n_0}/\Pi \rfloor$ and $q_{min} = v \cdot \Pi$ and $q_{max} = (v+1) \cdot \Pi$, with $b_{min} = b_{max} = 0$; this is a valid parameter choice for the Joye-Paillier algorithm. Table 1 shows that for a 1024-bit RSA modulus, our side channel attack enables to factor a fraction $\delta \simeq 8.4 \cdot 10^{-4}$ of the RSA moduli. If the algorithm is run only once inside a smart-card, this means that in practice a fraction $8.4 \cdot 10^{-4}$ of the smart-cards could be broken.

RSA	n_0	r	E[X]	δ
512 bits	256	43	18.7	$1.8 \cdot 10^{-3}$
768 bits	384	60	26.1	$1.1 \cdot 10^{-3}$
1024 bits	512	75	33.3	$8.4 \cdot 10^{-4}$
2048 bits	1024	131	60.0	$3.7 \cdot 10^{-4}$

Table 1. RSA modulus bit-size, bit-size n_0 of primes p and q, number r of primes p_i in Π, average number $E[X]$ of primality tests, and fraction δ of weak RSA moduli.

We note that the assumption $b_{min} = b_{max} = 0$ could be relaxed to small (known) values for b_{min} and b_{max}; in this case the value $b \in [b_{min}, b_{max}]$ would be exhaustively searched [4].

4 Countermeasures

In this section we discuss three possible countermeasures. Our first countermeasure is to periodically re-generate a fresh seed k so that the attacker doesn't

[4] Alternatively, one could try to derive a variant of Coppersmith's theorem for primes of the form $q = 2^n \cdot x + b \cdot \Pi + C$, with unknown x and b, and known constants Π and C.

obtain enough information about the prime q. Let $s > 0$ be the maximum number of primality tests performed for a given seed k. The integer s should be small enough to prevent the previous attack, but not too small to keep the same level of efficiency; we propose to take $s = n_0/4$ where n_0 is the prime bit-size. One gets the modified algorithm :

Modified Prime Generation Algorithm :
Parameters : s, Π odd, b_{min}, b_{max}, v
Output : a random prime $q \in [q_{min}, q_{max}]$

1. Compute $\ell \leftarrow v\Pi$
2. Let $i \leftarrow 0$.
3. Randomly choose $k \in (\mathbb{Z}/\Pi\mathbb{Z})^*$
4. Randomly choose $b \in \{b_{min}, \dots, b_{max}\}$ and set $t \leftarrow b\Pi$
5. Set $q \leftarrow k + t + \ell$
6. If (q `even`) then $q \leftarrow \Pi - k + t + \ell$
7. If $T(q) =$ `false` then
   (a) Set $k \leftarrow 2k \mod \Pi$
   (b) Let $i \leftarrow i + 1$
   (c) If $i < s$ then go to step 5, otherwise go to step 2.
8. Output q

Our second countermeasure consists in replacing the instruction $k \leftarrow 2 \cdot k \mod \Pi$ by $k \leftarrow 2^t \cdot k \mod \Pi$ where t is a small random integer. If $k \leftarrow 2^t \cdot k \mod \Pi$ is implemented in constant time, then the attacker doesn't know for which integers $k_i = 2^i \cdot a \mod \Pi$ he gets the parity bits, which prevents the previous attack. This second countermeasure is efficient because the additional running time is probably negligible compared to the primality test running time.

Our third countermeasure is to implement the test so that it doesn't leak. The standard way is to compute both sides of the branch and select the correct result. This countermeasure is analogous to the *square & multiply always* and *double & add always* countermeasures in RSA exponentiation and ECC scalar multiplication (see [2]). The instruction :

6. If (q `even`) then $q \leftarrow \Pi - k + t + \ell$

is then replaced by :

6. (a) Let $u \leftarrow q \mod 2$
   (b) Set $A[0] \leftarrow \Pi - k + t + \ell$
   (c) Set $A[1] \leftarrow q$
   (d) Let $q \leftarrow A[u]$

In this case the computation of $\Pi - k + t + \ell$ is always performed so it is likely more difficult for the attacker to recover the parity of q.

5 Conclusion and Open Problem

We have demonstrated that an improper implementation of the Joye-Paillier prime generation algorithm may succumb to side-channel analysis. Our attack is based on the assumption that a Simple Power Analysis can give us the parity bits which appear in a branch condition. We have shown that for certain parameters ($b_{min} = b_{max} = 0$) and for 1024-bit RSA moduli, a fraction 10^{-3} of those moduli can be factored efficiently. However in practice, some of the parity bits obtained through SPA may be erroneous. Therefore an interesting open problem is to find an attack that works with errors; formally :

Open Problem: let $\beta > 0$, let Π be an odd integer and let $k_0 \in \mathbb{Z}_{\Pi}$. Define the sequence $k_i = 2^i\, k_0 \bmod \Pi$ and $b_i = k_i \mod 2$ for $i \geq 1$. Let $b'_i = b_i$ with probability $1 - \beta$ and $b'_i = 1 - b_i$ otherwise, independently for each $i \geq 1$. Given the sequence b'_i for $i \geq 1$, find k_0 in time polynomial in $\log \Pi$.

Acknowledgments : we wish to thank Marc Joye and the anonymous reviewers for their valuable comments.

References

1. D. Coppersmith, *Small solutions to polynomial equations, and low exponent vulnerabilities*. J. of Cryptology, 10(4)233-260, 1997. Revised version of two articles of Eurocrypt '96.
2. J.S. Coron, *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*. Proceedings of CHES '99. LNCS, Springer-Verlag.
3. W. Dupuy and S. Kunz-Jacques, *Resistance of Randomized Projective Coordinates Against Power Analysis*, Proceedings of CHES 2005, LNCS.
4. P.A. Fouque, S. Kunz-Jacques, G. Martinet, F. Muller, and F. Valette. *Power Attack on Small RSA Public Exponent*. Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '06), LNCS 4249, Springer-Verlag, 2006.
5. M. Joye, P. Paillier and S. Vaudenay, *Efficient Generation of Prime Numbers*. Proceedings of CHES'00, Lecture Notes in Computer Science, Springer-Verlag No 1965, pp. 340-354.
6. M. Joye and P. Paillier, *Fast Generation of Prime Numbers of Portable Devices: An Update*. Proceedings of CHES 2006, LNCS 4249, pp. 160-173, 2006.
7. P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science, vol. 1666, pp. 388–397, Springer-Verlag, 1999.
8. G. Miller, *Riemann's Hypothesis and Tests for Primality*. J. Comp. Syst. Sci. 13, 300-317, 1976.