Side Channel Analysis of Random Prime Generation

Christophe Clavier

University of Limoges

Master 2 Cryptis

Problem Statement

Random Prime Generation

Given $\ell > 0$, generate at random an ℓ-bit (hopefully) prime integer q with possibly several goals in mind:

- Efficiency on constrained environment
- Randomness quality (entropy, distribution)
- Negligible probability of being composite (for probable primes)
- Implementation secure w.r.t. side-channel threats

Provable versus Probable

- Generating provable primes is often considered as being a too hard task on embedded devices: intricate implementation, too slow,...
- Hopefully efficient probable primality tests exist (Fermat, Miller-Rabin) that are quite suitable in these environments

Université de Limoges

A Simple Naive Method

Given a primality test $\top$ (Fermat, Miller-Rabin,...), one can simply generate an ℓ-bit prime q in the following way:

Naive Prime Generation

1: pick a random ℓ-bit odd integer q
2: **if** $\top(q) =$ `false` **then**
3: go to step 1
4: **return** q

Advantage:

- Generated primes are uniformly distributed

Drawbacks:

- Requires the availability of many random bits at each iteration
- Non optimized expected number of trials: $\ln(2^\ell)/2 = 0.347\ell$
  (177 trials on average for 512-bit primes)

Université de Limoges

Naive Incremental Method

Naive Incremental Prime Generation

1: pick a random ℓ-bit odd integer q
2: **while** $\top(q) = \texttt{false}$ **do**
3: $\quad q \leftarrow q + 2$
4: **return** q

- Random bits are only required for the initial candidate (neglecting randomness needed for executing $\top$)
- Output distribution is no more strictly uniform (it does not seem easy to exploit this slight bias)
- Expected number of trials is unchanged

Université de Limoges

Using a Basic Sieve

It is possible to decrease the number of calls to the expensive primality test by previously ruling out candidates that are divisible by a small prime p with $p \in \{p_1 = 2, p_2 = 3, \ldots, p_k\}$

Incremental Prime Generation with a Basic Sieve

1: pick a random ℓ-bit odd integer q
2: $j \leftarrow 2$ [p_j is the first j^{th} prime]
3: **while** $j \leq k$ **do**
4: **if** p_j divides q **then**
5: $q \leftarrow q + 2$ and go to step 2
6: $j \leftarrow j + 1$
7: **if** $\top(q) =$ `false` **then**
8: $q \leftarrow q + 2$ and go to step 2
9: **return** q

- Each candidate tested by $\top$ has a probability to be prime $\frac{\pi}{\phi(\pi)}$ times larger than an arbitrary ℓ-bit random integer (with $\pi = \prod_{j=1}^{j=k} p_j$)
- The expected number of primality tests is reduced in the same proportion
  (e.g. 61 trials on average for 512-bit primes with $k = 8$, $p_k = 19$)

Université de Limoges

Optimizing the Sieve

- It is possible to make the sieving phase more efficient by avoiding long integer divisions at each iteration.
- The residues $w_j = q \bmod p_j$ are stored and updated at each iteration. If any of them is zero then consider a new candidate.

Incremental Prime Generation with an Optimized Sieve

1: pick a random ℓ-bit odd integer q
2: **for** $j = 2$ **to** k **do**
3: $\quad w_j \leftarrow q \bmod p_j$ [initialization of residues]
4: **while** $w_j = 0$ for some $j \in \{2, \ldots, k\}$ **do**
5: $\quad$ **for** $j = 2$ **to** k **do**
6: $\quad\quad w_j \leftarrow (w_j + 2) \bmod p_j$ [update of residues]
7: $\quad q \leftarrow q + 2$
8: **if** $\top(q) =$ `false` **then**
9: $\quad$ **for** $j = 2$ **to** k **do**
10: $\quad\quad w_j \leftarrow (w_j + 2) \bmod p_j$ [update of residues]
11: $\quad q \leftarrow q + 2$ and go to step 4
12: **return** q

- Randomness requirement, output distribution and expected number of trials are unchanged

Université de Limoges

The Joye-Paillier Method [JPV00,JP06]

This first published prime generation method dedicated to On-Board Key Generation presents several advantages:

- All candidates are coprime with a product of primes π that is nearly as large as the targeted bit-length ℓ $\rightarrow$ the set of sieving small primes is optimal.
- When a candidate is updated, it automatically remains coprime with π $\rightarrow$ the sieving process is implicitly performed at virtually no cost.
- The primes can be generated in any arbitrary interval $[q_{min}, q_{max}]$.

The algorithm requires $\pi = \prod_j p_j$ (with $p_j \neq 2$), and integers b_{min}, b_{max} satisfying:

- (P1) $1 - \varepsilon < \frac{(b_{max} - b_{min} + 1)\pi}{q_{max} - q_{min} + 1} \leq 1$
- (P2) $b_{min}\pi \geq q_{min}$
- (P3) $b_{max}\pi + \pi - 1 \leq q_{max}$
- (P4) $\frac{\Phi(\pi)}{\pi}$ as small as possible

The Joye-Paillier Method [JPV00,JP06]

Joye-Paillier Prime Generation

Input: π odd, b_{min}, b_{max}
Output: a random prime $q \in [q_{min}, q_{max}]$

1: $k \xleftarrow{\$} (\mathbb{Z}/\pi\mathbb{Z})^\star$
2: $b \xleftarrow{\$} \{b_{min}, \ldots, b_{max}\}$
3: $t \leftarrow b\pi$
4: $q \leftarrow t + k$
5: **if** q is even **then**
6: $\quad q \leftarrow t + (\pi - k)$
7: **if** $\top(q) = \texttt{false}$ **then**
8: $\quad k \leftarrow 2k \bmod \pi$ and go to step 4
9: **return** q

- Two efficient methods are provided that generate units at step 1.
- After b has been randomly chosen at step 2, all candidates q are generated in $[b\pi, b\pi + \pi - 1]$ and are coprime with 2π.
- The candidate update at step 8 is quite efficient.

Université de Limoges

The Fouque-Tibouchi Method [FT11]

Fouque and Tibouchi devised a prime generation method that simultaneously fulfils two target properties:

- Output distribution is hardly distinguishable from the uniform distribution over the primes. This is clearly not the case for the incremental search, nor for Joye-Paillier.
- Requires a small amount of randomness bits. The simple naive method (uniform distribution) requires as many as ℓ bits of randomness for each candidate.

Fouque-Tibouchi versus Joye-Paillier:

- Candidates are systematically (at no cost) coprime with a large product of small primes $\pi \quad \rightarrow$ as fast as Joye-Paillier.
- Dual of Joye-Paillier in that it fixes the residue modulo π, and freely modifies the highest significant part. (Joye-Paillier fixes the generation interval $[b\pi, b\pi + \pi - 1]$ (highest bits) and freely modifies k the residue modulo π.)

Université de Limoges

The Fouque-Tibouchi Method [FT11]

Fouque-Tibouchi Prime Generation

Input: a small integer $u \ll \ell$ (typically $u = 32$), $\pi \approx 2^{\ell-u}$ a product of small primes
Output: a (quite close to) uniformly distributed ℓ-bit random prime q

1: $b \xleftarrow{\$} (\mathbb{Z}/\pi\mathbb{Z})^\star$ [using Joye-Paillier unit generation]
2: **repeat**
3: $\quad a \xleftarrow{\$} \{0, \ldots, \lfloor \frac{2^\ell}{\pi} \rfloor - 1\}$
4: $\quad q \leftarrow a\pi + b$
5: **until** $\top(q) = \texttt{true}$
6: **return** q

- Requires only u bits of randomness at each iteration
- As fast as Joye-Paillier while achieving a uniform distribution
- Successive candidates are independent from each others

Université de Limoges

Provable Prime Generation Method [CFTP12]

- All previously prime generation methods are based on probabilistic primality tests (e.g. Miller-Rabin)
- $\Rightarrow$ there exists a tiny probability that it outputs a composite integer
- Known *provable* primality tests (AKS, ECPP) are by far too expensive and not suitable for embedded implementation

A Constructive Approach

- Instead of *testing* random numbers for primality, one can think to constructively generate prime numbers "from scratch"
- Given a prime of size ℓ bits, a variation of Pocklington's theorem is used to infer a new prime number of size up to 2ℓ bits
- An iterative application of this principle allows to generate primes of arbitrary size

Université de Limoges

Provable Prime Generation Method [CFTP12]

Pocklington's theorem

Let $n > 3$ be an odd integer, and let $n = rF + 1$ where the factorization of F is known as $F = \prod_{j=1}^{s} q_j^{e_j}$. If there exists an integer a such that

(i) $a^{n-1} \equiv 1 \pmod{n}$ and

(ii) $\gcd(a^{(n-1)/q_j} - 1, n) = 1$ for each $j = 1 \ldots s$,

then every prime divisor p of n is congruent to 1 modulo F.
In particular, if $F > \sqrt{n} - 1$ then n is prime.

- Cannot be used on any given integer as it requires the factorization of $n - 1$ to be partially known.

Provable Prime Generation Method [CFTP12]

Particular Case of Pocklington's theorem

Let p be an odd prime, and r an integer such that $r < p$. Let $n = 2rp + 1$.

(i) If there exists an integer a with $2 \leq a < n$ such that $a^{n-1} \equiv 1 \pmod{n}$ and $\gcd(a^{2r} - 1, n) = 1$ then n is prime.

(ii) If n is prime, the probability that a random value a satisfies $a^{n-1} \equiv 1 \pmod{n}$ and $\gcd(a^{2r} - 1, n) = 1$ is $1 - 1/p$.

- A generation algorithm can be derived by iteratively producing provable primes twice larger at each iteration: $p_{i+1} = 2 \cdot r_i \cdot p_i + 1$
- Example for 512-bit primes: 17 bits $\rightarrow$ 33 $\rightarrow$ 65 $\rightarrow$ 129 $\rightarrow$ 257 $\rightarrow$ 512 bits

- Advantage: generation of provable primes on embedded devices as efficient as existing probabilistic methods
- Drawback: output distribution entropy is not optimal (e.g. 467 bits instead of 503 bits of entropy for 512-bit primes)

Université de Limoges

Complexity of Modular Exponentiation

Complexity of Modular Exponentiation

Let $n > 0$ a modulus, $k > 0$ an exponent and $a, b \in \mathbb{Z}_n$ two operands:

addition $a + b \bmod n$ can be computed in time $\mathcal{O}(\log n)$

multiplication $a \times b \bmod n$ can be computed in time $\mathcal{O}(\log^2 n)$

exponentiation $a^k \bmod n$ is computed as a sequence of $\mathcal{O}(\log k)$ multiplications modulo $n \Rightarrow$ it can be computed in time $\mathcal{O}(\log k \log^2 n)$

RSA decryption The exponent is full-size ($\log k \approx \log n$) $\Rightarrow c^d \bmod n$ is computed in time $\mathcal{O}(\log^3 n)$ (even in CRT mode)

RSA encryption The exponent is a small fixed size integer $\Rightarrow m^e \bmod n$ is computed in time $\mathcal{O}(\log^2 n)$

Université de Limoges

Complexity of Prime Generation

Complexity of Prime Generation

- All prime generation are based upon a series of primality tests
- Testing q with either Fermat or Miller-Rabin requires a few full-size exponentiations modulo $q \Rightarrow \top(q)$ is computed in $\mathcal{O}(\log^3 q)$
- As the average number of tested candidates is $\mathcal{O}(\log q)$, generating a ℓ-bit prime q is done in time $\mathcal{O}(\log^4 q)$, that is $\mathcal{O}(\ell^4)$

Université de Limoges

Only Half of the Bits are Enough

All side-channel attacks that will be presented share the property that they allows to recover the value of the generated prime q modulo a large known integer (either a power of 2 or the product of many small primes).

A result from Coppersmith, which allows to find small roots of bivariate polynomials, shows that q can be recovered as soon as this modulus is only slightly greater than $\ell/2$ bits.

Consequently, to recover a 512-bit prime (for 1024-bit RSA keys) it is only sufficient to recover 256 bits of modular information.

Université de Limoges

Simple Power Analysis on the Joye-Paillier Method [CC07]

- At each iteration of the Joye-Paillier prime generation, a parity test is performed on the new candidate:

  ...
  $k \leftarrow 2k \bmod \pi$
  $q \leftarrow t + k$
  if (q is even)
  $q \leftarrow t + (\pi - k)$

- If this test is observable by SPA, then one can recover as many (least significant) bits of information on the generated prime than the number of tests performed since the generation of the initial candidate.
- $\Rightarrow$ long sequences of candidates before detecting a prime may expose sufficiently many bits of q (or p) to factorize $n = pq$.
- Allows to factorize a fraction of about 10^{-3} of the generated 1024-bit moduli.

Université de Limoges

Simple Power Analysis on the Joye-Paillier Method [CC07]

Suggested countermeasures:

1. periodically refresh the seed k (e.g. every $\ell/4$ iterations) in order to expose not sufficiently many bits about q,
2. replace
   $$k \leftarrow 2k \bmod \pi$$
   by
   $$k \leftarrow 2^{\alpha} k \bmod \pi$$
   with α a small random integer,
3. implement the parity test so that it does not leak:
   - $q \leftarrow t + k$
   - $u \leftarrow q \bmod 2$
   - $A[0] \leftarrow t + (\pi - k) + t$
   - $A[1] \leftarrow q$
   - $q \leftarrow A[u]$

Université de Limoges

Simple Power Analysis on the Incremental Method with Basic Sieve [FGS09]

Target Algorithm

1: pick a random ℓ-bit odd integer q
2: $j \leftarrow 2$ [p_j is the first j^{th} prime]
3: **while** $j \leq k$ **do**
4: **if** p_j divides q **then**
5: $q \leftarrow q + 2$ and go to step 2
6: $j \leftarrow j + 1$
7: **if** $\top(q) =$ `false` **then**
8: $q \leftarrow q + 2$ and go to step 2
9: **return** q

- Let $q_0, \ldots, q_m$ be the sequence of successive candidates that are processed by the sieve ($q_m = q$ is the output prime).
- For each q_i the attacker is supposed to be able to detect by SPA which small prime p_j is the first to divide q_i (or that no such p_j exists in the case where $\top$ is executed).

Université de Limoges

Simple Power Analysis on the Incremental Method with Basic Sieve [FGS09]

- Note that $q = q_m = q_i + 2(m - i)$.
  If p_j happens to divide q_i then:

$$q \equiv 2(m - i) \pmod{p_j}$$

- Combining such congruences leads (by CRT) to the knowledge of q modulo the product of all p_j known to divide any q_i.
- In the case of RSA key generation, same kind of information can be collected from the generation of p and combined with the information about q.
- $n = pq$ can be factorized if whole modular information is greater than $\ell/2$ bits.

Université de Limoges

Simple Power Analysis on the Incremental Method with Basic Sieve [FGS09]

Improving the probability of success:

- If the number of bits of modular information is slightly not sufficient, one may consider a few primes p_j for which we did not detect that p_j divides some q_i.
- For such p_j, we may have detected that p_j does not divide some q_i. For all these i's we know that :

$$q \not\equiv 2(m - i) \pmod{p_j}$$

- It is then possible to exhaust all remaining guesses on q mod p_j and try to factorize with this extra modular information.

Empirical results

This attack can factorize about 10% of 1024-bit RSA moduli if the sieve considers $k = 54$ small primes ($p_{54} = 251$), and about 25% of them if $k = 70$.

Université de Limoges

Horizontal Correlation Power Analysis on the Incremental Method with Optimized Sieve [BJL+14]

Target Implementation

- Sieve on first 54 primes (fit on a single byte)
- The current candidate is represented by an array of residues modulo each small prime: $w_j = q \bmod p_j$
- Updating the candidate (e.g. $q \leftarrow q + 2$) amounts to update the whole array of residues : $w_j \leftarrow w_j + 2 \bmod p_j$
  $\rightarrow$ only reductions on 8-bit integers

Attack Assumption

- One is able to locate side-channel trace segments corresponding to any sieve update operation $w_j \leftarrow w_j + 2 \bmod p_j$

Université de Limoges

Horizontal Correlation Power Analysis on the Incremental Method with Optimized Sieve [BJL+14]

- Let $q_{i,j} = q_i \bmod p_j$ denote the residue of the i^{th} candidate modulo the j^{th} prime.
- Notice that all $q_{i,j}$ are dependent from each others through the relation

$$q_{i,j} \equiv q - 2(m - i) \pmod{p_j}$$

  where only q is unknown.
- Any guess on $q \bmod p_j$ leads to a sequence of predicted residues $\{q_{i,j}\}_i$.
- Horizontal CPA allows to check the consistency between this sequence of residues and the observed leakages.
- If m is large enough then $q \bmod p_j$ can be recovered for a sufficiently large number of small primes so that the bitlength of their product is greater than $\ell/2$.

Université de Limoges

Horizontal Correlation Power Analysis on the Incremental Method with Optimized Sieve [BJL+14]

Simulation results

- Success rate for recovering 512-bit primes is greater than 75 % when the noise standard deviation is $\sigma = 1$ and about 17,5 % when $\sigma = 3$.
- Notice that the attack can not recover 1024-bit primes if only 8-bit small primes are used (since $\log_2(\prod_{j=1}^{j=54} p_j) \approx 335 < 512$ bits).

Proposed countermeasures

- Adding dummy w_j updates, or shuffling their order of computation.

Recommendation

The authors recommend to avoid prime generation methods where the successive candidates are computed in a deterministic way from an initial seed. Recommended: Fouque-Tibouchi or provable prime construction [CFTP12].

Conclusion

- Not so many side-channel attacks proposed so far on prime generation methods.
- There is probably room for many new side-channel attack ideas.
- No practical fault attack has ever been published.
- Research on both SCA and FA on prime generation is encouraged.
- Probably, non-deterministic candidate updates should be preferred.

Université de Limoges