Fast Generation of Prime Numbers on Portable Devices

An Update

Marc Joye

Thomson Security Labs
marc.joye@thomson.net

(Joint Work with Pascal Paillier)

Outline

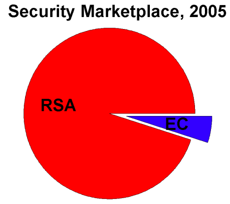
The Need for Prime Numbers

Our Algorithms

Analysis

Conclusion

RSA Is Everywhere



- RSA = 95% of security products
  - Alternative technology: elliptic curves
- RSA comes in many standards

  Encryption PKCS #1 (RSA-OAEP), IEEE P1363a
  Signature PKCS #1 (PSS/PSS-R), ISO/IEC 9796 (RW), ANSI X9.31, NIST/FIPS PUB 186-2, ITU-T X.509
- RSA has been impacting smart-card technologies for 10 years
  - Each and every chip manufacturer proposes its own cryptoprocessor(s) = specific hardware design(s)
  - Designing a cryptoprocessor = huge investments
    - financially
    - technologically (heavy devs, strong patents)
  - RSA performances are critical for all PK-enabled smart cards

THOMSON images & beyond

RSA In Practice

Key generation

1. Generate 2 large primes p and q (e.g., of 1024 bits)
   - $\gcd(e, p-1) = \gcd(e, q-1) = 1$ with $e = 3$ or $2^{16}+1$, etc
2. Obtain
   - public key: $N = p\,q$ and e
   - private key: (p, q)

Signing a message

- Padding: $\mathtt{msg} \longmapsto \mu(\mathtt{msg})$
- Signature: $S = \mu(\mathtt{msg})^d \bmod N$ where $d = e^{-1} \bmod (p-1)(q-1)$

Verification Given $\mathtt{msg}$ and S, check whether $S^e \bmod N = \mu(\mathtt{msg})$

Signature scheme $\Longrightarrow$ authentication, integrity, non-repudiation

THOMSON images & beyond

RSA Key Generation

Main step (complicated...)

- On input $(\mathrm{random}, \ell, e)$, construct

$$q \leftarrow \texttt{GenPrime}(\mathrm{random}, \ell, e)$$

  - Invoke this twice to get p, q

Key derivation functions (easy)

- On input (e, p, q), compute
  - $N = p\,q$
  - $\begin{cases} d = e^{-1} \bmod (p-1)(q-1) & \text{(STD mode)} \\ d_p, d_q, i_q & \text{(CRT mode)} \end{cases}$

Off-Board/On-Board Key Generation

Off-board = keys generated in perso

- This is less secure for the end customer
- No dynamic control of key sizes, no re-generation

On-board key generation

- More secure for the end customer
- Re-generation on demand, dynamically-chosen sizes
- Applications can manage keys on their own
- Opens the way to key compression
  - e.g., 1024-bit RSA key $\mapsto$ 20 bytes

Specification of GenPrime

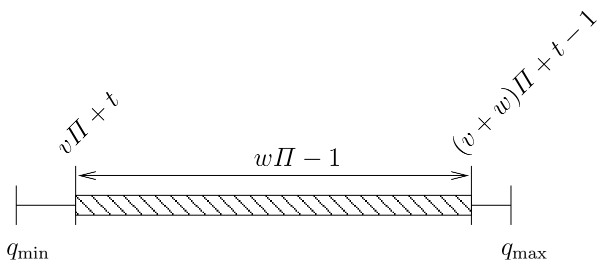
A prime number q generated by GenPrime in such that

1. q is an ℓ-bit number for a given bitsize ℓ
2. q belongs to $[q_{\min}, q_{\max}]$, e.g., $q_{\min} = \lceil 2^{\ell-1/2} \rceil$ and $q_{\max} = 2^\ell$
3. $\gcd(q-1, e) = 1$ where e is given

Also,

1. ℓ has a granularity of 1 bit, e.g., with $\ell \in [128, \ldots, 2048]$
2. GenPrime is pseudo-random: takes as input a random seed
3. GenPrime can integrate customizable constraints on the generated prime such as
   - Rabin-Williams primes
   - DSA primes (160-bit q, $q \mid p-1$)
   - standard ANSI X9.31 primes ($u \mid p-1$ and $s \mid p+1$)
   - strong primes ($u|p-1$, $s|p+1$ and $t|u-1$)

Choice of Parameters



- The prime candidates lie in

$$[v\Pi + t, (v+w)\Pi + t - 1] \subseteq [q_{\min}, q_{\max}]$$

- The prime candidates are automically coprime to

$$\Pi = \prod p_i$$

$\Longrightarrow \phi(\Pi)/\Pi$ as small as possible

GenPrime

Parameters: Π, t, v, w and $a \in \mathbb{Z}_m^* \setminus \{1\}$
Output: a random prime $q \in [q_{\min}, q_{\max}]$

1. Compute $l \leftarrow v\Pi$ and $m \leftarrow w\Pi$
2. Choose $k \in_R \mathbb{Z}_m^*$
3. Set $q \leftarrow [(k - t) \bmod m] + t + l$
4. If $(\mathsf{T}(q) = \texttt{false})$ then
   - 4.1 Set $k \leftarrow a \cdot k \pmod m$
   - 4.2 Go to Step 3
5. Output q

$$\left.\begin{array}{l} q \bmod \Pi \equiv [k - t] + t + 0 \equiv k \pmod{\Pi} \\ k^{(\mathrm{new})} = a \cdot k^{(\mathrm{old})} \in \mathbb{Z}_m^* \implies k^{(\mathrm{new})} \in \mathbb{Z}_\Pi^* \end{array}\right\} \implies \gcd(q, \Pi) = 1$$

GenPrime 2

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

1. Compute $l \leftarrow v\Pi$
2. Choose $k \in_R \mathbb{Z}_\Pi^*$
3. Choose $b \in_R \{b_{\min}, \ldots, b_{\max}\}$ and set $t \leftarrow b\,\Pi$
4. Set $q \leftarrow \begin{cases} k + t + l \\ (\Pi - k) + t + l \end{cases}$ (q is odd)
5. If $(\mathsf{T}(q) = \texttt{false})$ then
   - 5.1 Set $k \leftarrow 2k \pmod{\Pi}$
   - 5.2 Go to Step 4
6. Output q

Generation of Units

Proposition

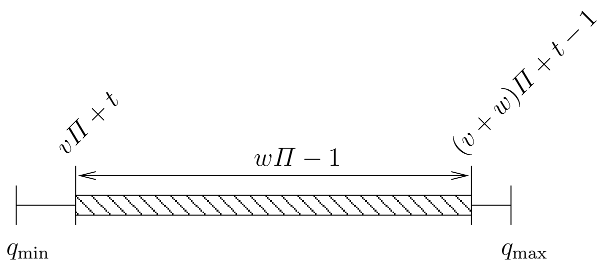
Let k, r be integers modulo m and assume $\gcd(r, k, m) = 1$. Then

$$k \leftarrow [k + r(1 - k^{\lambda(m)}) \bmod m] \in \mathbb{Z}_m^*$$

Algorithm

1. `Randomly choose` $k \in [1, m[$
2. `Set` $U \leftarrow (1 - k^{\lambda(m)}) \bmod m$
3. `If` $(U \neq 0)$ `then`
   - 3.1 `Choose a random` $r \in [1, m[$
   - 3.2 `Set` $k \leftarrow k + rU \pmod{m}$ ["self-correctness"]
   - 3.3 `Go to Step 2`
4. `Return` $k \in \mathbb{Z}_m^*$

Length Extendability



- Parameters of `GenPrime` are
  - (Π, t, v, w)
  - $\lambda(m)$ with $m = w\Pi$
- Heavily depend $q_{\min} = \lceil 2^{\ell_0 - 1/2} \rceil$ and $q_{\max} = 2^{\ell_0}$

Scalability

Our algorithms allow to use the parameters sized for ℓ_0 to generate primes of bitsize $\ell \geqslant \ell_0$

THOMSON images & beyond

RSA Primes

An RSA prime q must satisfy $\gcd(e, q-1) = 1$

Arbitrary public exponent e

- The test $\gcd(e, q-1) = 1$ should be explicitly added

"Small" public exponent e

- Let $e = \prod_i {e_i}^{\nu_i}$
- If $e_i \mid \Pi$ for all i then our algorithms can be adapted such that the condition $\gcd(e, q-1) = 1$ is automatically satisfied
  - This includes the popular choices $e = 3$ or $e = 17$

Safe/Quasi-Safe Primes

A safe prime q is such that $(q-1)/2$ is also a prime

[More generally, a d-quasi-safe prime q is such that $(q-1)/2^d$ is also a prime]

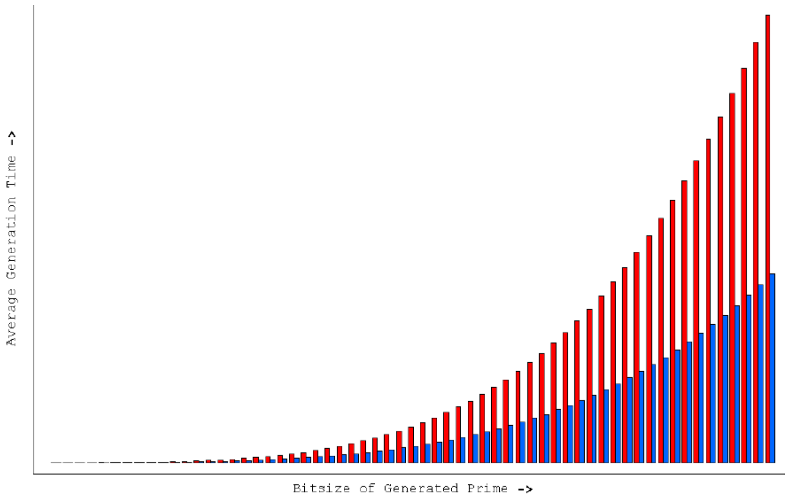
Modified search sequence

Our algorithms can be adapted such that every candidate q is coprime to Π but also $(q-1)/2$ is coprime to Π

Performance Analysis

Average number of primality tests for generating q

Bitsize ℓ	256	384	512	768	1024
$H[\ell, 10]$	28.03	42.04	56.05	84.08	112.1
GenPrime[ℓ]	18.72	26.12	33.29	46.90	59.98



Security Properties

GenPrime has nearly maximal entropy

$$H_{\max} - H < \frac{1-\gamma}{\ln 2} = 0.609949$$

Bitsize ℓ	256	384	512	768	1024
$H_{\max}$	246.767	374.179	501.762	757.176	1012.76
H	246.194	373.596	501.173	756.581	1012.16
$H_{\max} - H$	0.572795	0.583093	0.588773	0.594834	0.598092

GenPrime has negligible collision probalility

Bitsize ℓ	256	384	512	1024
$\nu \leqslant$	$3.30 \cdot 10^{-152}$	$4.28 \cdot 10^{-229}$	$4.93 \cdot 10^{-306}$	$5.49 \cdot 10^{-614}$

Summary

- Improved techniques
  - better performances than previously suggested algorithms
  - reduced statistical deviation (generation of units)
- Extended capabilities
  - length extendability
  - RSA condition automatically satisfied for “small” e
- Safe primes and quasi-safe primes
  - modified search sequence
- “Provably” reliable algorithms
  - excellent output distribution
  - negligible collision probability