Introducción a la Factorización De Semiprimos (RSA)

Normalmente, en este foro hablamos de la criptografía estándar, aquella que corre en nuestros dispositivos. Muchos sabréis que algunos de los esquemas criptográficos basados en clave pública o en una única clave (simétricos) sufrirán debilidades cuando el cuántico se torne realidad.

Esto es debido a que problemas como la factorización de enteros, el cálculo del logaritmo discreto, ya sea el aplicado a curvas elípticas o en finite fields podrán ser resueltos. Una respuesta obvia sería aumentar el tamaño de clave en la parametrización de estos esquemas, pero implicaría hacerlo cada vez que la computación cuántica avance.

Hoy vamos a explicar una forma de factorizar números compuestos por dos primos, es decir, semiprimos. Es un problema conocido pues nos daría la habilidad de romper esquemas como RSA. Nótese que RSA también presenta el RSA Problem y está basado en calcular raices e-ésimas mod n.

Antes de empezar, quiero decir que el lector tiene que tener un buen conocimiento en matemáticas: tanto en teoría de números como en algebra lineal.

Factorización De Fermat (Punto Medio)

Sea $N = p \cdot q$ tal que $p, q \in \mathbb{P}$ es un producto de dos primos, por lo tanto N es semiprimo.

Queremos encontrar los factores p, q que componen el módulo utilizado por una clave pública RSA. Esto nos permitiría generar firmas digitales válidas, además de intervenir en el TLS Handshake ya sea falsificando valores cuando RSA se usa para firma, o bien, descifrando la PMK (Pre Master Key) lo cual nos permitiría hacer un Man In The Middle en plano, como antiguamente 

Hace unos años, cuando empecé en esto, mi problema favorito era el de la factorización de enteros, curiosamente, la de semiprimos. Resulta que reinventé la rueda al descubrir que entre dos primos siempre existe un punto medio r el cual equidista con misma distancia d sobre los primos p, q Si traducimos a cristiano esto sería:

$(r + d) \cdot (r - d) = n$ donde $r = \frac{p+q}{2}$ y $d = \frac{p-q}{2}$ Si vemos más allá nos damos cuenta que $(x + y) \cdot (x - y) = x^2 - y^2 = n$ lo que nos da la expresión $r^2 - d^2 = p \cdot q = n$ ¿Bonito verdad? ¿Pero cómo aplicarlo?

Un ejemplo sería $p = 17, q = 11, n = 17 \cdot 11 = 187$ entonces $r = \frac{17+11}{2} = 14$ y $d = 17 - 14 = 3$ Si nos imaginamos los enteros positivos alineados en una línea horizontal, vemos que 14 equidista 3 posiciones de $17, 11$.

Para usarlo, al saber que $r^2 - d^2 = n$ hacemos $\sqrt{n - r^2} = d^2$ desde $r = \sqrt{n} + 1$ hasta dar con una solución entera. Pena que a Pierre de Fermat, se le ocurriera hace casi 400 años, conociéndose este método por Fermat Factorization.

En el mundo real nunca sabremos p, q de antemano, sólo n y nuestro interés se centra en encontrar los valores r, d mediante una técnica llamada Quadratic Sieve o Criba Cuadrática.

Quadratic Sieve : Primera Fase

En teoría de números un número b-smooth es aquel que tiene factores primos menores o iguales que b por ejemplo $30 = 2 \cdot 3 \cdot 5$ es 5-smooth pero no 3-smooth. Establecemos $b \in \mathbb{P}$ como primo y seguimos nuestro análisis.

Si tomamos la congruencia $x^2 \equiv y^2 \pmod{N}$ entonces $x^2 - y^2 \equiv 0 \pmod{N} \rightarrow x^2 - y^2 \mid N$ (divide a N) por lo tanto $(x + y) \cdot (x - y) \mid N$ y tanto $x + y$ como $x - y$ son factores de N.

Empezamos tomando valores y de la congruencia $x^2 \equiv y \pmod{N}$ Para eso vamos generando valores x Expresamos y en la base prima b-smooth $y = p_1^{e_1} \ldots, p_b^{e_b} = \prod_{i=1}^{b} p_i^{e_i}$.

Guardamos y reducimos mod 2 los exponentes de la base prima de cada residuo y en una matriz de exponentes $M_e \in F_2^{b \times k}$ donde k indica el número de congruencias y que hemos obtenido de la criba. En cada columna guardamos los exponentes e_i de cada congruencia $x_i^2 \equiv y_i \pmod{N}$ Entonces vemos M_e como:

$$M_e = \begin{bmatrix} e_{1,1}, \ldots, e_{k,1} \\ \vdots \ldots \vdots \\ e_{1,b} \ldots e_{k,b} \end{bmatrix}$$

Segunda Fase

¿Por qué hemos reducidos los exponentes e_i modulo 2 y encima en columnas? Tened en cuenta que si multiplicamos dos números, sus exponentes se suman. Si tras realizar la multiplicación todos los exponentes son pares/even, entonces $a \cdot b$ tiene raíz cuadrada entera. Si los exponentes de a, b son $\alpha = (\alpha_1, \ldots, \alpha_b), \beta = (\beta_1, \ldots, \beta_b)$ entonces la multiplicación resulta en $a \cdot b = \prod_{i=1}^{b} p_i^{\alpha_i + \beta_i}$, si cada sumando $\alpha_i + \beta_i$ es par, la tupla modulo 2 resultado de $\alpha + \beta$ daría 0 en cada posición.

De esta forma, encontramos residuos b-smooth cuyo producto resulta en una tupla de exponentes enteramente par, y como estamos en $\mathbb{F}_2$ obtenemos una tupla que consiste de 0's y de longitud b por lo tanto $\alpha + \beta = (0, \ldots, 0)_b$.

Si revisáis los apuntes de Álgebra Lineal, el rango de una matriz nos indica la dimensión de la imagen. Una matriz se toma como una transformación lineal, cuenta con dominio y codominio. Como nuestra matriz tiene dimensión $b \times k$ entonces el sistema $M_e \overrightarrow{x} = \overrightarrow{0}$ nos daría en el vector $\overrightarrow{x} = (x_1, \ldots, x_k)$ la ansiada combinación lineal, puesto que una matriz por un vector, no es más que la suma de una combinación lineal de sus columnas. Las ecuaciones de tipo $A \cdot \overrightarrow{x} = \overrightarrow{0}$ se resuelven calculando el kernel o nullspace de la matriz A Si tenemos una matriz $m \times n$ con $m > n$ entonces $rk(A) \leq n$ por lo tanto $Dim(A) = n$ y $Dim(A) - rk(A) = n - r = Dim(Ker(A))$.

Por lo tanto la matriz que representa el kernel de M_e tiene dimensión $k \times (k - rk(A)$ Si el rango de M_e no baja de $\min(b, k)$ la única solución es $\overrightarrow{x} = (0, \ldots, 0)_k$ llamada solución trivial. Para hallar soluciones no triviales el rango ha de ser inferior a $\min(b, k)$ y para esto, como las columnas de M_e corresponden a los exponentes de cada residuo, si el rango baja sabemos que existe un producto de residuos que tiene raíz entera. De esta forma, encontramos una combinación lineal $\sigma = (\sigma_1, \ldots, \sigma_k)$ sobre las columnas que representan los exponentes, es decir, una suma finita de estas columnas (modulo 2) que da como solución una tupla de longitud b con todos los elementos a 0 Entonces $M_e \cdot \sigma = \overrightarrow{0}_b$ tal y como deseábamos.

Tercera Fase

Ahora, vamos a encontrar los valores a, b tal que $a^2 - b^2 = (a + b)(a - b) = N$ Como hemos guardado todos los valores x_i, y_i tal que $x_i^2 \equiv y \pmod{N}$ haciendo uso de la tupla σ calculamos el producto de los valores $x_i = \sigma_i$ en a Lo mismo para los valores de y_i y lo hacemos en b^2 Entonces $b = \sqrt{b^2}$ y $\gcd(a \pm b, N) \mid N$.

La fase de criba finalizaría al dar con el kernel de M_e Como podéis ver es una fase muy amplía, donde necesitamos muchas congruencias para encontrar una combinación lineal de los exponentes que sea par. Además es necesario en todo momento almacenar las listas de las congruencias, tanto los x_i como los y_i Todo ello mediante una representación matricial, donde cada columna corresponde a los exponentes de cada termino y_i en la base b-smooth. Para concluir, vemos como el kernel nos permite extraer valores para los cuales existe una combinación líneas de las tuplas de exponentes que da cero. De esta forma sabemos que valores y_i forman un cuadrado mediante su producto o multiplicación.

Ejemplo Quadratic Sieve

Vamos a poner un pequeño ejemplo para ilustrar la teoría anteriormente expuesta.

Tomamos $p = 4663, q = 3581, N = 16698203$ la base está compuesta de los 50 primeros primos entonces es 229-smooth es decir

$B = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109,$
$113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229\}$

Todos los pares x_i, y_i que tratemos tienen que ser 229-smooth, es decir, factorizables entre productos de los primos en la base B.

Si empezamos con $x = \lfloor \sqrt{N} \rfloor + 1$ tras 11 iteraciones, tenemos la lista
$X = (4107, 4114, 4125, 4371, 4401, 4657, 4745, 4836, 5217, 5221, 5355)$ y la de residuos cuadráticos
$Y = (169246, 226793, 317422, 2407438, 2670598, 4989446, 5816822, 6688693, 10518886, 10560638, 11977822)$

La matriz de exponentes M_e tiene dimensión 50×11 porque cada columna tiene longitud $b = 50$ por lo tanto 50 filas y al haber 11 congruencias, de ahí sale. Nos enfrentamos al sistema $M_e \cdot \overrightarrow{x} = \overrightarrow{0}_{50}$ Queremos una suma de columnas en M_e que nos de completamente cero, y de esta forma sabremos que productos $y_i y_j$ dan un cuadrado mod N. La matriz M_e se muestra a continuación:

$$
\begin{pmatrix}
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
$$

La matriz M_e tiene rango 10 modulo 2. Por lo tanto, su kernel o nullspace tiene dimensión 1 y es representable mediante una matriz 11×1 en este caso $nsM = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ Entonces, la ecuación $M_e \cdot x = 0 \rightarrow A \cdot (nsM \cdot c) = 0$ tiene dos soluciones (una de ellas la trivial) porque la dimensión es 1, por lo tanto una tupla en $\mathbb{F}_2$ tiene 2 valores, en este caso 0 ó 1. Por lo tanto fijamos $\sigma = (0, 0, 1, 0, 0, 0, 0, 0, 0, 1)$ y $A\sigma = 0$ Comprobad vosotros mismos que la tercera columna y la onceava columna de la matriz M_e suman 0 mod 2 en cada coordenada/posición.

Por lo tanto, sabemos que de la lista X multiplicamos los valores 3º y el 11º, esto es, $a = 4125 \cdot 5355 = 22089375$ y lo mismo con la lista Y, nos queda $b^2 = 317422 \cdot 11977822 = 3802024214884$ Calculamos $\sqrt{b^2} = \sqrt{3802024214884} = 1949878$ Entonces $\gcd(a + b, N) = \gcd(22089375 + 1949878, N) = 3581$ y $\gcd(a - b, N) = \gcd(22089375 - 1949878, N) = 4633]$.

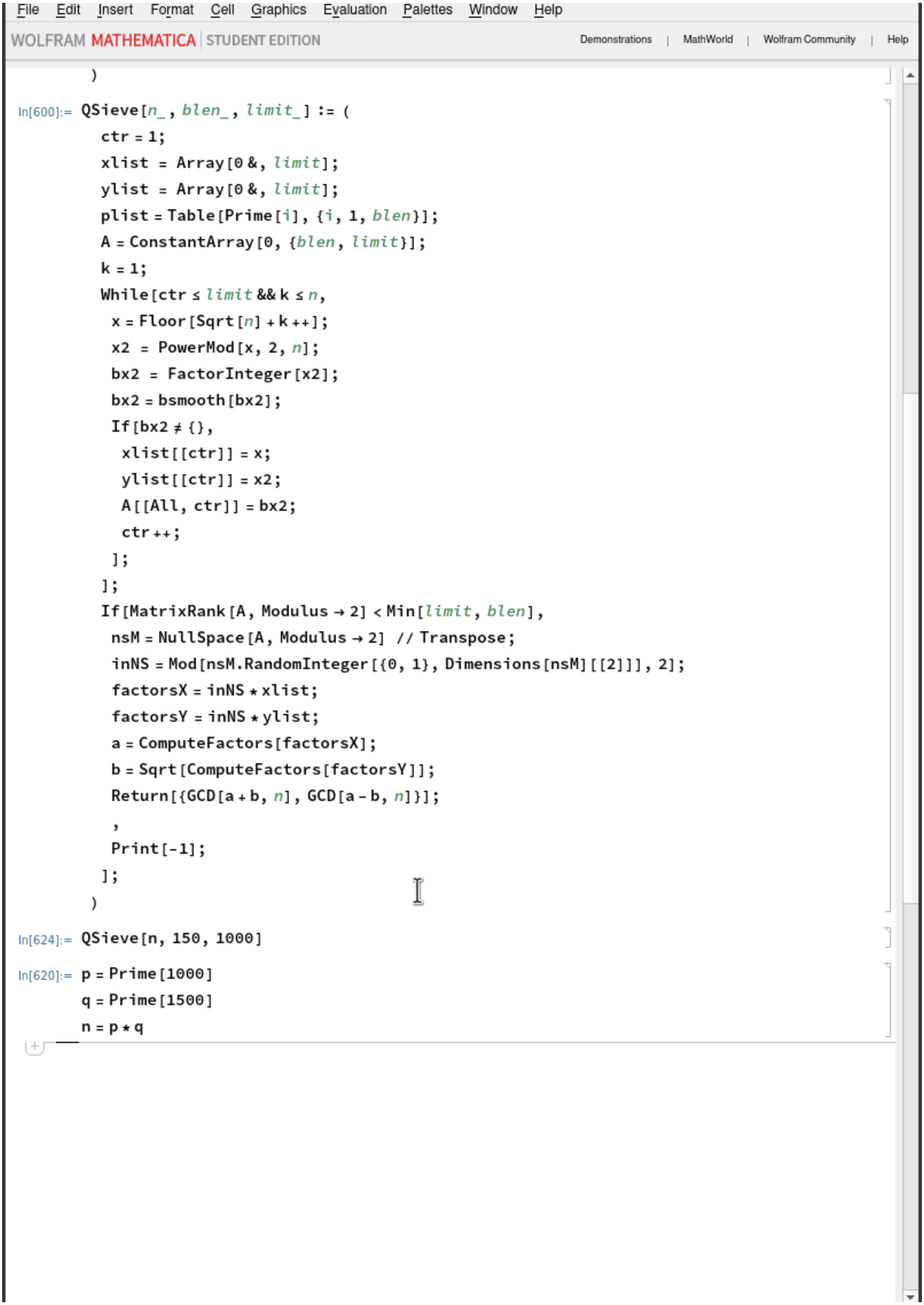
Aquí os dejo el código del algoritmo en Mathematica. Sé que lo puedo mejorar, lo he hecho en un par de horas para colgarlo junto a este tuto. En C++ utilizando la lib de GMP es igual de sencillo, mientras tengas las matemáticas de tu lado, todo va bien.

```
bsmooth[x_] := (
   exps = Array[0 &, Length[plist]];
   found = False;
   i = 1;
   While[found == False && i <= Length[x],
    pos = Position[plist, x[[i]][[1]]] ;
    If[pos == {},
     found = True;
     ,
     pos = pos[[1]];
     exps[[pos]] = Mod[x[[i]][[2]], 2];
     ];
    i++;
    ];
   If[found == True,
    Return[{}];
    ,
    Return[exps];
    ];
   );

ComputeFactors[list_] := (
  k = 1;
  Do[If[list[[i]] != 0, k = k*list[[i]]], {i, 1, Length[list]}];
  Return[k];
  )

QSieve[n_, blen_, limit_] := (
  ctr = 1;
  xlist = Array[0 &, limit];
  ylist = Array[0 &, limit];
  plist = Table[Prime[i], {i, 1, blen}];
  A = ConstantArray[0, {blen, limit}];
  k = 1;
  While[ctr <= limit && k <= n,
   x = Floor[Sqrt[n] + k++];
   x2 = PowerMod[x, 2, n];
   bx2 = FactorInteger[x2];
   bx2 = bsmooth[bx2];
   If[bx2 != {},
    xlist[[ctr]] = x;
    ylist[[ctr]] = x2;
    A[[All, ctr]] = bx2;
    ctr++;
    ];
   ];
  If[MatrixRank[A, Modulus -> 2] < Min[limit, blen],
   nsM = NullSpace[A, Modulus -> 2] // Transpose;
   inNS = Mod[nsM.RandomInteger[{0, 1}, Dimensions[nsM][[2]]], 2];
   factorsX = inNS*xlist;
```

```
    factorsY = inNS*ylist;
    a = ComputeFactors[factorsX];
    b = Sqrt[ComputeFactors[factorsY]];
    Return[{GCD[a + b, n], GCD[a - b, n]}];
    ,
    Print[-1];
    ];
  )
```

Para llamar a la función, se necesita el semiprimo n, un número blen que indica la longitud de la base de primos b-smooth y un límite, cuantas congruencias ha de recolectar. Después de recolectar, intenta obtener una solución en la transpuesta del nullspace de la matriz de exponentes. El resto es componer los factores y hacer el GCD.

He capturado un .GIF para mostrar un ejemplo del algoritmo corriendo en Mathematica:



Vemos que el nullspace aquí ya no es 11×1 sino 1000×869 Obviamente, el ejemplo que he puesto, era facilito pero el código trabaja para valores más altos.

Y nada, ha quedado demostrado que la criba cuadrática encuentra una solución mientras que seamos capaces de factorizar números en una base b-smooth. Además de necesitar una combinación lineal válida que nos asegure la obtención de un cuadrado mediante producto de residuos cuadráticos.