Discrete-log attacks
and factorization
Part II
Tanja Lange
Technische Universiteit Eindhoven
14 June 2019

with some slides by
Daniel J. Bernstein

Q sieve
Sieving small integers $i > 0$
using primes 2, 3, 5, 7:

| 1 | 
|---|--|
| 2 | 2 |
| 3 | 3 |
| 4 | 2 2 |
| 5 | 5 |
| 6 | 2 3 |
| 7 | 7 |
| 8 | 2 2 2 |
| 9 | 3 3 |
| 10 | 2 5 |
| 11 | |
| 12 | 2 2 3 |
| 13 | |
| 14 | 2 7 |
| 15 | 3 5 |
| 16 | 2 2 2 |
| 17 | |
| 18 | 2 3 3 |
| 19 | |
| 20 | 2 2 5 |
|   | etc. |
Discrete-log attacks
and factorization
Part II
Tanja Lange
Technische Universiteit Eindhoven
14 June 2019
with some slides by
Daniel J. Bernstein

Q sieve

Sieving small integers $i > 0$
using primes 2, 3, 5, 7:

\[
\begin{array}{c|cccc}
1 & 2 & 2 \\
2 & 3 \\
3 & 2 & 2 \\
4 & 3 & 5 \\
5 & 2 & 3 \\
6 & 7 \\
7 & 2 & 2 \\
8 & 2 & 2 \\
9 & 3 & 3 \\
10 & 5 \\
11 \\
12 & 2 & 2 \\
13 \\
14 & 2 & 7 \\
15 & 3 & 5 \\
16 & 2 & 2 \\
17 \\
18 & 2 & 3 & 3 \\
19 \\
20 & 2 & 2 & 5 \\
\end{array}
\]

etc.

\[
\begin{array}{c|cccc}
1 & 2 & 2 \\
2 & 3 \\
3 & 2 & 2 \\
4 & 3 & 5 \\
5 & 2 & 3 \\
6 & 7 \\
7 & 2 & 2 \\
8 & 2 & 2 \\
9 & 3 & 3 \\
10 & 5 \\
11 \\
12 & 2 & 2 \\
13 \\
14 & 2 & 7 \\
15 & 3 & 5 \\
16 & 2 & 2 \\
17 \\
18 & 2 & 3 & 3 \\
19 \\
20 & 2 & 2 \\
\end{array}
\]

etc.
Q sieve

Sieving small integers $i > 0$
using primes 2, 3, 5, 7:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

etc.

Q sieve

Sieving $i$ and 611 + $i$ for small $i$
using primes 2, 3, 5, 7:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>11</th>
<th>13</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td>11</td>
<td>13</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td></td>
<td>7</td>
<td>13</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>3</td>
<td>5</td>
<td>13</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5</td>
<td></td>
<td>13</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

etc.
**Q sieve**

Sieving small integers $i > 0$ using primes 2, 3, 5, 7:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

etc.

**Q sieve**

Sieving $i$ and $611 + i$ for small $i$ using primes 2, 3, 5, 7:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|612| 2 | 2 | 3 | 3 |
|613|   |   |   |   |
|614| 2 |   |   |   |
|615|   |   | 3 | 5 |
|616| 2 | 2 | 2 |   |
|617|   |   |   |   |
|618| 2 | 3 |   |   |
|619|   |   |   |   |
|620| 2 | 2 |   | 5 |
|621|   |   | 3 | 3 | 3 |
|622| 2 |   |   |   |
|623|   |   |   |   |
|624| 2 | 2 | 2 | 2 | 3 |
|625|   |   |   | 5 |   |
|626| 2 |   |   |   |
|627| 3 |   |   |   |
|628| 2 | 2 |   |   |
|629|   |   |   |   |
|630| 2 | 3 | 3 | 5 |   |
|631|   |   |   |   |

etc.
Q sieve

Sieving small integers $i > 0$ using primes 2, 3, 5, 7:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>2 2 2</td>
<td>2 2 2 3</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>3 3</td>
<td>3 3</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>2 5</td>
<td>2 5</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>2 3</td>
<td>2 3</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>2 2 3</td>
<td>2 2 4</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>2 2</td>
<td>2 2</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>2 7</td>
<td>2 7</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>3 5</td>
<td>3 5</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>2 2</td>
<td>2 2</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>2 2 2</td>
<td>2 2 2 2</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>3 3</td>
<td>3 3</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>2 3</td>
<td>2 3</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>2 2</td>
<td>2 2</td>
</tr>
<tr>
<td>20</td>
<td>etc.</td>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>
Q sieve

Sieving $i > 0$ small integers using primes $2, 3, 5, 7$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>12</td>
<td>2</td>
<td>13</td>
<td>14</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>612</th>
<th>613</th>
<th>614</th>
<th>615</th>
<th>616</th>
<th>617</th>
<th>618</th>
<th>619</th>
<th>620</th>
<th>621</th>
<th>622</th>
<th>623</th>
<th>624</th>
<th>625</th>
<th>626</th>
<th>627</th>
<th>628</th>
<th>629</th>
<th>630</th>
<th>631</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

etc.

Have complete factorization of the “congruences” $i$ $(611 + i)$ for some $i$'s.

$14 \cdot 625 = 2^1 \cdot 3^0 \cdot 5^5 \cdot 7^4$.

$64 \cdot 675 = 2^6 \cdot 3^3 \cdot 5^2 \cdot 7^0$.

$75 \cdot 686 = 2^1 \cdot 3^1 \cdot 5^2 \cdot 7^3$.

$14 \cdot 64 \cdot 75 - \overline{2^4 \cdot 3^2 \cdot 5^4 \cdot 7^2} = (2^4 \cdot 3^2 \cdot 5^4 \cdot 7^2)^2$.

$\gcd \{611, 14 \cdot 64 \cdot 75 - \overline{2^4 \cdot 3^2 \cdot 5^4 \cdot 7^2}\} = 47$.

$611 = 47 \cdot 13$. 
Q sieve

Sieving $i$ and $611 + i$ for small $i$ using primes 2, 3, 5, 7:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$612$</th>
<th>$613$</th>
<th>$614$</th>
<th>$615$</th>
<th>$616$</th>
<th>$617$</th>
<th>$618$</th>
<th>$619$</th>
<th>$620$</th>
<th>$621$</th>
<th>$622$</th>
<th>$623$</th>
<th>$624$</th>
<th>$625$</th>
<th>$626$</th>
<th>$627$</th>
<th>$628$</th>
<th>$629$</th>
<th>$630$</th>
<th>$631$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

etc.

Have complete factorization of the “congruences” $i \cdot (611 + i)$ for some $i$’s.

$14 \cdot 625 = 2^1 \cdot 3^0 \cdot 5^4 \cdot 7^1$

$64 \cdot 675 = 2^6 \cdot 3^3 \cdot 5^2 \cdot 7^0$

$75 \cdot 686 = 2^1 \cdot 3^1 \cdot 5^1 \cdot 7^2$

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686$

$= 2^8 \cdot 3^4 \cdot 5^8 \cdot 7^4 = (2^4 \cdot 3^2 \cdot 5^4 \cdot 7^2)^2$

$\gcd\{611, 14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686\} = 47.$

$611 = 47 \cdot 13.$
Q sieve

Sieving $i$ and $611 + i$ for small $i$ using primes 2, 3, 5, 7:

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

etc.

Have complete factorization of the “congruences” $i(611 + i)$ for some $i$’s.

$14 \cdot 625 = 2^{14}3^05^47^1$.

$64 \cdot 675 = 2^63^35^27^0$.

$75 \cdot 686 = 2^13^15^27^3$.

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 = 2^83^45^87^4 = (2^43^25^47^2)^2$.

$\gcd\{611, 14 \cdot 64 \cdot 75 - 2^43^25^47^2\} = 47$.

$611 = 47 \cdot 13$. 
Q sieve

Sieving \(i\) and \(611 + i\) for small \(i\) using primes 2, 3, 5, 7:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

etc.

Have complete factorization of the “congruences” \(i(611 + i)\) for some \(i\)’s.

\[
14 \cdot 625 = 2^1 3^0 5^4 7^1.
\]

\[
64 \cdot 675 = 2^6 3^3 5^2 7^0.
\]

\[
75 \cdot 686 = 2^1 3^1 5^2 7^3.
\]

\[
14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 = 2^8 3^4 5^8 7^4 = (2^4 3^{2} 5^{4} 7^2)^2.
\]

\[
gcd\{611, 14 \cdot 64 \cdot 75 - 2^4 3^{2} 5^{4} 7^2\} = 47.
\]

\[
611 = 47 \cdot 13.
\]
Sieve $i$ and $611 + i$ for small $i$ using primes $2, 3, 5, 7$:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>612</td>
<td>2 2 3 3</td>
<td></td>
</tr>
<tr>
<td>613</td>
<td></td>
<td></td>
</tr>
<tr>
<td>614</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>615</td>
<td>2 2 2 3 5</td>
<td></td>
</tr>
<tr>
<td>616</td>
<td>2 2 2 7</td>
<td></td>
</tr>
<tr>
<td>617</td>
<td></td>
<td></td>
</tr>
<tr>
<td>618</td>
<td>2 3</td>
<td></td>
</tr>
<tr>
<td>619</td>
<td></td>
<td></td>
</tr>
<tr>
<td>620</td>
<td>2 2 5</td>
<td></td>
</tr>
<tr>
<td>621</td>
<td>3 3 3</td>
<td></td>
</tr>
<tr>
<td>622</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>623</td>
<td>2 2 2 2 3 7</td>
<td></td>
</tr>
<tr>
<td>624</td>
<td>2 2 2 3</td>
<td></td>
</tr>
<tr>
<td>625</td>
<td>2 2 2 2 3</td>
<td></td>
</tr>
<tr>
<td>626</td>
<td>2 2 3</td>
<td></td>
</tr>
<tr>
<td>627</td>
<td>2 2</td>
<td></td>
</tr>
<tr>
<td>628</td>
<td>2 2</td>
<td></td>
</tr>
<tr>
<td>629</td>
<td>2 3 3 5 7</td>
<td></td>
</tr>
<tr>
<td>630</td>
<td>2 3 3 5 7</td>
<td></td>
</tr>
</tbody>
</table>

Have complete factorization of the “congruences” $i(611 + i)$ for some $i$'s.

$14 \cdot 625 = 2^1 3^0 5^4 7^1$.  
$64 \cdot 675 = 2^6 3^3 5^2 7^0$.  
$75 \cdot 686 = 2^1 3^1 5^2 7^3$.

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 = 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2$.  
$\gcd\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\} = 47$.

$611 = 47 \cdot 13$.

Why did this find a factor of $611$?
Was it just blind luck:  
$\gcd\{611, \text{random}\} = 47$?
No.
By construction $611$ divides $s - t$ where $s = 14 \cdot 64 \cdot 75$ and $t = 2^4 3^2 5^4 7^2$.
So each prime $> 7$ dividing $611$ divides either $s - t$ or $s + t$.

Not terribly surprising (but not guaranteed in advance!) that one prime divides $s - t$ and the other divides $s + t$.
Have complete factorization of the “congruences” \( i(611 + i) \) for some \( i \)’s.

\[
\begin{align*}
14 \cdot 625 &= 2^13^05^47^1. \\
64 \cdot 675 &= 2^63^35^27^0. \\
75 \cdot 686 &= 2^13^15^27^3. \\
14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 &= 2^83^45^87^4 = (2^43^25^47^2)^2. \\
gcd\{611, 14 \cdot 64 \cdot 75 - 2^43^25^47^2\} &= 47. \\
611 &= 47 \cdot 13.
\end{align*}
\]

Why did this find a factor of 611? Was it just blind luck: \( \gcd\{611, \text{random}\} = 47 \)?

No.

By construction 611 divides \( s - t \) where \( s = 14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 \) and \( t = 2^43^25^47^2 \).

So each prime \( > 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).
Have complete factorization of the “congruences” $i(611 + i)$ for some $i$’s.

$14 \cdot 625 = 2^1 3^0 5^4 7^1$.
$64 \cdot 675 = 2^6 3^3 5^2 7^0$.
$75 \cdot 686 = 2^1 3^1 5^2 7^3$.

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686$
$= 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2$.

$\text{gcd}\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\}$
$= 47$.

$611 = 47 \cdot 13$.

Why did this find a factor of 611? Was it just blind luck: $\text{gcd}\{611, \text{random}\} = 47$?

No.

By construction 611 divides $s - t$ where $s = 14 \cdot 64 \cdot 75$ and $t = 2^4 3^2 5^4 7^2$.

So each prime $> 7$ dividing 611 divides either $s - t$ or $s + t$.

Not terribly surprising (but not guaranteed in advance!) that one prime divided $s - t$ and the other divided $s + t$. 
Have complete factorization of the “congruences” \( i(611 + i) \) for some \( i \)'s.

\[
14 \cdot 625 = 2^1 3^0 5^4 7^1.
\]
\[
64 \cdot 675 = 2^6 3^3 5^2 7^0.
\]
\[
75 \cdot 686 = 2^1 3^1 5^2 7^3.
\]

\[
14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 = 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2.
\]

\[
gcd\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\} = 47.
\]

\[
611 = 47 \cdot 13.
\]

Why did this find a factor of 611? Was it just blind luck: \( \gcd\{611, \text{random}\} = 47 \)?

No.

By construction 611 divides \( s^2 - t^2 \) where \( s = 14 \cdot 64 \cdot 75 \) and \( t = 2^4 3^2 5^4 7^2 \).

So each prime \( > 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).
Have complete factorization of the "congruences" \( i(611 + i) \) for some \( i \)’s.

\[
\begin{align*}
14 \cdot 625 &= 2^{13} 5^{4} 7^{1} \\
64 \cdot 675 &= 2^{6} 3^{3} 5^{2} 7^{0} \\
75 \cdot 686 &= 2^{1} 3^{1} 5^{2} 7^{3}
\end{align*}
\]

\[
611 = 47 \cdot 13.
\]

Why did this find a factor of 611? Was it just blind luck:
\[
gcd\{611, \text{random}\} = 47?
\]

No.

By construction 611 divides \( s^2 - t^2 \)
where \( s = 14 \cdot 64 \cdot 75 \) and \( t = 2^{4} 3^{2} 5^{4} 7^{2} \).

So each prime > 7 dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!)
that one prime divided \( s - t \) and the other divided \( s + t \).

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors \((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\) happened to have sum 0 mod 2.

But we didn’t need this luck!
Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

Why did this find a factor of 611? Was it just blind luck:
\[
gcd\{611, \text{random}\} = 47?
\]

No.

By construction 611 divides \( s^2 - t^2 \)
where \( s = 14 \cdot 64 \cdot 75 \) and \( t = 2^{4} 3^{2} 5^{4} 7^{2} \).

So each prime > 7 dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!)
that one prime divided \( s - t \) and the other divided \( s + t \).
Why did this find a factor of 611? Was it just blind luck: \( \gcd \{611, \text{random} \} = 47? \)

No.

By construction 611 divides \( s^2 - t^2 \) where \( s = 14 \cdot 64 \cdot 75 \) and \( t = 2^4 \cdot 3^2 \cdot 5^4 \cdot 7^2 \).

So each prime \( \geq 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors \((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\) happened to have sum 0 mod 2.

But we didn’t need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.
Why did this find a factor of 611? Was it just blind luck: \( \gcd\{611, \text{random}\} = 47? \)

No.

By construction 611 divides \( s^2 - t^2 \) where \( s = 14 \cdot 64 \cdot 75 \) and \( t = 2^43^25^47^2 \).

So each prime \( > 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors \((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\) happened to have sum 0 mod 2.

But we didn’t need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.
Why did this find a factor of 611?
Was it just blind luck:
gcd\{611, \text{random}\} = 47?
No.

By construction 611 divides \(s^2 - t^2\)
where \(s = 14 \cdot 64 \cdot 75\)
and \(t = 2^43^25^47^2\).
So each prime > 7 dividing 611 divides either \(s - t\) or \(s + t\).

Not terribly surprising (but not guaranteed in advance!)
that one prime divided \(s - t\)
and the other divided \(s + t\).

Why did the first three completely factored congruences have square product?
Was it just blind luck?
Yes. The exponent vectors 
\((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\)
happened to have sum 0 mod 2.

But we didn’t need this luck!
Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.
Why did this find a factor of 611? Was it just blind luck: \( \text{gcd}\{611; \text{random}\} = 47\)?

By construction 611 divides \( s^2 - t^2 \) where \( s = 14 \cdot 64 \cdot 75 \cdot 2^4 3^2 5^4 7^2 \).

Any prime \( > 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!)

Yes. The exponent vectors \((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\) happened to have sum 0 mod 2.

But we didn’t need this luck!

Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over \( \mathbb{F}_2 \).

Guaranteed to find subsequence if number of vectors exceeds length of each vector.

E.g. for \( n = 671 \):

\begin{align*}
1(n + 1) & = 2^5 3^1 5^0 7^1, \\
4(n + 4) & = 2^2 3^3 5^2 7^0, \\
15(n + 15) & = 2^1 3^1 5^1 7^3, \\
49(n + 49) & = 2^4 3^2 5^1 7^2.
\end{align*}

\( \mathbb{F}_2 \)-kernel of exponent matrix is generated by \((0, 1, 0, 1, 1)\) and \((1, 0, 1, 1, 0)\);

E.g., \( 1(n + 1)15(n + 15)49(n + 49) \) is a square.
Why did this find a factor of 611? Was it just blind luck: \( \gcd(611; \text{random}) = 47 \)?

No. By construction 611 divides \( s^2 - t^2 \) where 
\[
s = 14 \cdot 64 \cdot 75 \\
t = 2^4 3^2 5^4 7^2 
\]
11 divides \( s^2 - t^2 \) or \( s + t \). Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors 
\[
(1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)
\]
happened to have sum 0 mod 2.

But we didn’t need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over \( \mathbb{F}_2 \). Guaranteed to find subsequence if number of vectors exceeds length of each vector.

E.g. for \( n = 671 \): 
\[
1(n + 1) = 2^5 3^1 \\
4(n + 4) = 2^2 3^2 \\
15(n + 15) = 2^1 3^3 \\
49(n + 49) = 2^4 3^2 \\
64(n + 64) = 2^6 3^1
\]
\( \mathbb{F}_2 \)-kernel of exponent matrix is gen by \((0 \ 1 \ 0 \ 1 \ 1) \) and \((1 \ 0 \ 1 \ 1 \ 0) \); e.g., \( 1(n + 1)15(n + 15)49(n + 49) \) is a square.
Why did this find a factor of 611?

Was it just blind luck:
\[ \gcd(611; \text{random}) = 47? \]

No.

By construction 611 divides \( s^2 - t^2 \) where

\[ s = 14 \cdot 64 \cdot 75 \]

\[ t = 2^{14325472}. \]

So each prime \( > 7 \) dividing 611 divides either \( s - t \) or \( s + t \).

Not terribly surprising (but not guaranteed in advance!) that one prime divided \( s - t \) and the other divided \( s + t \).

Why did the first three completely factored congruences have square product?

Was it just blind luck?

Yes. The exponent vectors

\( (1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3) \)

happened to have sum 0 mod 2.

But we didn’t need this luck!

Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over \( \mathbb{F}_2 \).

Guaranteed to find subsequence if number of vectors exceeds length of each vector.

e.g. for \( n = 671 \):

\[ 1(n + 1) = 2^{5315071}; \]
\[ 4(n + 4) = 2^{235270}; \]
\[ 15(n + 15) = 2^{131517^3}; \]
\[ 49(n + 49) = 2^{432517^2}; \]
\[ 64(n + 64) = 2^{631517^2}. \]

\( \mathbb{F}_2 \)-kernel of exponent matrix is generated by \( (0 1 0 1 1) \) and \( (1 0 1 1 0) \),
e.g., \( 1(n + 1)15(n + 15)49(n + 49) \) is a square.
Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors 
\((1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\) happened to have sum 0 mod 2.

But we didn’t need this luck!
Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over \(F_2\).
Guaranteed to find subsequence if number of vectors exceeds length of each vector.

e.g. for \(n = 671\):
\[
\begin{align*}
1(n + 1) &= 2^53^15^07^1; \\
4(n + 4) &= 2^23^35^27^0; \\
15(n + 15) &= 2^13^15^17^3; \\
49(n + 49) &= 2^43^25^17^2; \\
64(n + 64) &= 2^63^15^17^2.
\end{align*}
\]

\(F_2\)-kernel of exponent matrix is gen by \((0 \ 1 \ 0 \ 1 \ 1)\) and \((1 \ 0 \ 1 \ 1 \ 0)\); e.g., \(1(n + 1)15(n + 15)49(n + 49)\) is a square.
Why did the first three completely factored congruences have square product? Was it just blind luck? Yes. The exponent vectors $(1, 0, 4, 1)$, $(6, 3, 2, 0)$, $(1, 1, 2, 3)$ happened to have sum $0 \mod 2$. But we didn’t need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum $0 \mod 2$.

This is linear algebra over $\mathbb{F}_2$. Guaranteed to find subsequence if number of vectors exceeds length of each vector. e.g. for $n = 671$:

- $1(n + 1) = 2^53^15^07^1$;
- $4(n + 4) = 2^23^55^27^0$;
- $15(n + 15) = 2^13^15^17^3$;
- $49(n + 49) = 2^43^25^17^2$;
- $64(n + 64) = 2^63^15^17^2$.

$\mathbb{F}_2$-kernel of exponent matrix is generated by $(0 \ 1 \ 0 \ 1 \ 1)$ and $(1 \ 0 \ 1 \ 1 \ 0)$; e.g.,

- $1(n + 1)15(n + 15)49(n + 49)$ is a square.

Plausible conjecture: \textipa{Q} sieve can separate the odd prime divisors of any $n$, not just 611. Given $n$ and parameter $y$:

Try to completely factor $i(n + i)$ for $i \in \mathbb{Z} \backslash \{0\}$ into products of primes $\leq y$.

Look for nonempty set $I$ of $i$'s with $i(n + i)$ completely factored and with $\prod_{i \in I} i(n + i)$ square.

Compute $\gcd\{n; s - t\}$ where $s = \prod_{i \in I} i$ and $t = \sum_{i \in I} (n + i)$. 

Why did the first three completely factored congruences have square product? Was it just blind luck? Yes. The exponent vectors 

\[(1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)\]

happened to have sum 0 mod 2. But we didn't need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over \( \mathbb{F}_2 \). Guaranteed to find subsequence if number of vectors exceeds length of each vector.

\[\text{e.g. for } n = 671:\]

\[1(n + 1) = 2^53^15^07^1;\]
\[4(n + 4) = 2^23^35^27^0;\]
\[15(n + 15) = 2^13^15^17^3;\]
\[49(n + 49) = 2^43^25^17^2;\]
\[64(n + 64) = 2^63^15^17^2.\]

\( \mathbb{F}_2 \)-kernel of exponent matrix is gen by \( (0 \ 1 \ 0 \ 1 \ 1) \) and \( (1 \ 0 \ 1 \ 1 \ 0) \); e.g., \(1(n + 1)15(n + 15)49(n + 49)\) is a square.

Plausible conjecture: \( Q \) sieve can separate the odd prime divisors of any \( n \), not just \( 611 \).

Given \( n \) and parameter \( y \):

Try to completely factor \( i(n + i) \) for \( i \in \{1, 2, 3, \ldots \} \) into products of primes \( \leq y \).

Look for nonempty subsequence \( i(n + i) \) with \( i \in I \) and with \( \prod i(n + i) \) square.

Compute \( \gcd\{n, s - t\} \) where \( s = \prod i \) and \( t = \sum i(n + i) \).
This is linear algebra over $\mathbb{F}_2$. Guaranteed to find subsequence if number of vectors exceeds length of each vector.

e.g. for $n = 671$:

$1(n + 1) = 2^53^15^07^1$;
$4(n + 4) = 2^23^35^27^0$;
$15(n + 15) = 2^13^15^17^3$;
$49(n + 49) = 2^43^25^17^2$;
$64(n + 64) = 2^63^15^17^2$.

$\mathbb{F}_2$-kernel of exponent matrix is gen by $(0\ 1\ 0\ 1\ 1)$ and $(1\ 0\ 1\ 1\ 0)$; e.g., $1(n + 1)15(n + 15)49(n + 49)$ is a square.

Plausible conjecture: $\mathbb{Q}$ sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

Try to completely factor $i(n + i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\leq y$.

Look for nonempty set $I$ of vectors, with $i(n + i)$ completely factored and with $\prod_{i \in I} i(n + i)$ square.

Compute $\gcd\{n, s - t\}$ where $s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n + i)}$. 
This is linear algebra over $\mathbb{F}_2$.
Guaranteed to find subsequence if number of vectors
exceeds length of each vector.
e.g. for $n = 671$:
$1(n + 1) = 2^5 3^1 5^0 7^1$;
$4(n + 4) = 2^2 3^3 5^2 7^0$;
$15(n + 15) = 2^1 3^1 5^1 7^3$;
$49(n + 49) = 2^4 3^2 5^1 7^2$;
$64(n + 64) = 2^6 3^1 5^1 7^2$.
$\mathbb{F}_2$-kernel of exponent matrix is
gen by $(0 1 0 1 1)$ and $(1 0 1 1 0)$; e.g., $1(n + 1)15(n + 15)49(n + 49)$
is a square.

Plausible conjecture: $\mathbb{Q}$ sieve can
separate the odd prime divisors
of any $n$, not just 611.

Given $n$ and parameter $y$:
Try to completely factor $i(n + i)$
for $i \in \{1, 2, 3, \ldots, y^2\}$
into products of primes $\leq y$.
Look for nonempty set $I$ of $i$'s
with $i(n + i)$ completely factored
and with $\prod_{i \in I} i(n + i)$ square.
Compute $\gcd\{n, s - t\}$ where
$s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n + i)}$. 


linear algebra over $\mathbb{F}_2$.

We need to find subsequence if number of vectors exceeds length of each vector.

For $n = 671$:
- $1(n + 1) = 2^53^15^07^1$;
- $4(n + 4) = 2^23^35^27^0$;
- $15(n + 15) = 2^13^15^17^3$;
- $49(n + 49) = 2^43^25^17^2$;
- $64(n + 64) = 2^63^15^17^2$.

F$_2$-kernel of exponent matrix is generated by $(0\ 1\ 0\ 1\ 1)$ and $(1\ 0\ 1\ 1\ 0)$; e.g., $1(n + 1)15(n + 15)49(n + 49)$ is a square.

Plausible conjecture: $\mathbb{Q}$ sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:
- Try to completely factor $i(n + i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\leq y$.
- Look for nonempty set $I$ of $i$'s with $i(n + i)$ completely factored and with $\prod_{i \in I} i(n + i)$ square.
- Compute $\gcd\{n, s - t\}$ where $s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n + i)}$.

How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture: $\mathbb{Q}$ sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u(1 + o(1))u^2$; here $o(1)$ is as $u \to \infty$. 
This is linear algebra over $F_2$. Guaranteed to find a subsequence if the number of vectors exceeds the length of each vector.

For $n = 671$:

- $1(n + 1) = 2^{51} \cdot 3^{15071} \cdot 5^{270} \cdot 15^{173}$
- $4(n + 4) = 2^{51} \cdot 3^{15071} \cdot 5^{270} \cdot 15^{173}$
- $15(n + 15) = 2^{51} \cdot 3^{15071} \cdot 5^{270} \cdot 15^{173}$
- $49(n + 49) = 2^{51} \cdot 3^{15071} \cdot 5^{270} \cdot 15^{173}$

$F_2$-kernel of exponent matrix is generated by $(0, 1, 0, 1, 1)$ and $(1, 0, 1, 1, 0)$; e.g., $1(n + 1)15(n + 15)49(n + 49)$ is a square.

Plausible conjecture: Q sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

- Try to completely factor $i(n + i)$ for $i \in \{1, 2, 3, \ldots, y\}$ into products of primes $\leq y$.
- Look for nonempty set $I$ of $i$'s with $i(n + i)$ completely factored and with $\prod_{i \in I} (n + i)$ square.
- Compute $s = \prod_{i \in I} i$ and $t = \prod_{i \in I} (n + i)$.
- Plausible conjecture: Q sieve succeeds with $y = \lceil n^{1/u} \rceil$ for all $n \geq u(1 + o(1)) n^{1/u}$ where $o(1)$ is as $u \to \infty$.

How large does $y$ have to be for this to find a square? Uniform random integer in $[1; n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$. Q sieve succeeds for this to find a square.
Plausible conjecture: $\mathbf{Q}$ sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

Try to completely factor $i(n+i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\leq y$.

Look for nonempty set $I$ of $i$'s with $i(n+i)$ completely factored and with $\prod_{i \in I} i(n+i)$ square.

Compute $\gcd\{n, s-t\}$ where $s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n+i)}$.

How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture: $\mathbf{Q}$ sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))u^2}$; here $o(1)$ is as $u \to \infty$. 
Plausible conjecture: Q sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

Try to completely factor $i(n + i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\leq y$.

Look for nonempty set $I$ of $i$’s with $i(n + i)$ completely factored and with $\prod_{i \in I} i(n + i)$ square.

Compute $\gcd\{n, s - t\}$ where $s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n + i)}$.

How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture:

Q sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))u^2}$; here $o(1)$ is as $u \to \infty$. 
Plausible conjecture: Q sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

- Completely factor $i(n + i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$
- Products of primes $\leq y$.

- For nonempty set $I$ of $i$’s
- $i(n + i)$ completely factored
- $\prod_{i \in I} i(n + i)$ square.

- Compute $\gcd\{n, s - t\}$ where
  - $s = \prod_{i \in I} i(n + i)$
  - $t = \sqrt{\prod_{i \in I} i(n + i)}$.

How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture:

- Q sieve succeeds with $y = \lfloor n^{1/u} \rfloor$
- for all $n \geq u^{(1+o(1))u^2}$;
- here $o(1)$ is as $u \to \infty$.

More generally, if $y \in \exp q \frac{\log n}{\log \log n}$,
conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$’s:
replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{(\frac{\log n}{\log \log n})^{2c}}$.

Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost.
So linear algebra never dominates when $y$ is chosen properly.
Plausible conjecture: Q sieve can separate the odd prime divisors of any n, not just 611.

Given n and parameter y:

Try to completely factor $i(n + i)$ for $i \in \mathbb{Z} \cap [1, y^2]$ into products of primes $\leq y$.

Look for nonempty set I of i’s with $i(n + i)$ completely factored and with $\prod_{i \in I} i(n + i)$ square.

Compute $\gcd\{n; s - t\}$ where $s = \prod_{i \in I} i$ and $t = \prod_{i \in I} i(n + i)$.

How large does y have to be for this to find a square?

Uniform random integer in [1, n] has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture: Q sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))u^2};$ here o(1) is as $u \to \infty$.

More generally, if $y \in \exp \left( \frac{1}{2c} + o(1) \right) \log n \log \log n$, conjectured y-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of i’s:

replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{\left( \frac{(c+1)^2+o(1)}{2c} \right) \log n}$.

Increasing c past 1 increases number of i’s but reduces linear-algebra cost.

So linear algebra never dominates when y is chosen properly.
Plausible conjecture: Q sieve can separate the odd prime divisors of any $n$, not just 611.

Given $n$ and parameter $y$:

Try to completely factor $i(\overline{n+i})$ for $i \in \mathbb{Z}^+$ into products of primes $\leq y$.

Look for nonempty set $I$ of $i$'s with $i(\overline{n+i})$ completely factored and with $\bigwedge_{i \in I} i(\overline{n+i})$ square.

Compute $\gcd\{n; s-t\}$ where $s = \bigwedge_{i \in I} i$ and $t = \bigwedge_{i \in I} i(\overline{n+i})$.

How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture: Q sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))u^2}$; here $o(1)$ is as $u \to \infty$.

More generally, if $y \in \exp \sqrt{\left(\frac{1}{2c} + o(1)\right) \log n \log \log n}$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$'s: replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{\left(\frac{(c+1)^2+o(1)}{2c}\right) \log n \log \log n}$.

Increasing $c$ past 1 increases number of $i$'s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.
How large does $y$ have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

Plausible conjecture: $Q$ sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))}u^2$; here $o(1)$ is as $u \to \infty$.

More generally, if $y \in \exp \Theta((c+1)^2+o(1)) \log n \log \log n$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$'s: replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{(\frac{1}{2c} + o(1)) \log n \log \log n}$.

Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost. 
So linear algebra never dominates when $y$ is chosen properly.
How large does $y$ have to be to find a square?

A uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly $u^{-u}$.

One conjecture:

Q sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u(1+o(1))u^2$.

Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

More generally, if $y \in \exp q \frac{1}{2c} \log n \log \log n$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$’s:

1. replace $y^2$ with $y^{c+1+o(1)} = \exp \frac{(c+1)^2+o(1)}{2c} \log n \log \log n$

More careful analysis:

- $n+i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$
- only $30\%$ chance for $i \approx y^2$

Can we select congruences to avoid this degradation?
More generally, if $y \in \exp \sqrt{\left(\frac{1}{2c} + o(1)\right) \log n \log \log n}$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$’s: replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{\left(\frac{(c+1)^2 + o(1)}{2c}\right) \log n \log \log n}$.

Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.
More generally, if $y \in \exp \sqrt{\left( \frac{1}{2c} + o(1) \right) \log n \log \log n}$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$'s: replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{\left( \frac{(c+1)^2 + o(1)}{2c} \right) \log n \log \log n}$.

Increasing $c$ past 1 increases number of $i$'s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2 n$ if $i \approx y^2$.

More careful analysis:
$n + i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$;
only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?
More generally, if $y \in \exp \sqrt{\left(\frac{1}{2c} + o(1)\right) \log n \log \log n}$, conjectured $y$-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of $i$’s: replace $y^2$ with $y^{c+1+o(1)} = \exp \sqrt{\left(\frac{(c+1)^2+o(1)}{2c}\right) \log n \log \log n}$.

Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows.

Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.

$\approx yn$ if $i \approx y$;

$\approx y^2n$ if $i \approx y^2$.

More careful analysis:

$n + i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$;

only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?
More generally, if $y \in \exp_{\frac{1}{2c} + o(1)} \log n \log \log n$, the expected $y$-smoothness chance is $1 = y^c + o(1)$.

Though smooth congruences are infeasible, changing the range of $i$'s: replace $y^2$ with $y^{c+1+o(1)} = \exp[(c+1)^2+o(1)] \log n \log \log n$.

Increasing $c$ past 1 increases number of $i$'s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2n$ if $i \approx y^2$.

More careful analysis:
$n + i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$,
only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?

Choose a prime $q$, square of large prime.
Choose an arithmetic progression of $i$'s:
where $q$ divides each $i(n + i)$.

e.g. progression $q - (n \mod q)$, $2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of generalized congruence $i(n + i) = q$ for $i$'s in this sublattice.
e.g. check whether $i; (n + i) = q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$'s.
Rare for $i$'s to overlap.
More generally, if $y \in \exp \left( \frac{1}{2} c \log n \log \log n \right)$, conjectured $y$-smoothness chance is $1 = \exp \left( \frac{1}{2} c \right) + o(1)$. Find enough smooth congruences by changing the range of $i$’s: replace $y^2$ with $y^2 + 1 + o(1) = \exp \left( \left( c + 1 \right) / 2 \right) + o(1)$. Increasing $c$ past 1 increases number of $i$’s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows. Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.

\[
\approx yn \text{ if } i \approx y; \\
\approx y^2 n \text{ if } i \approx y^2.
\]

More careful analysis:

$n + i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$, only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?

Choose $q$, square of large prime.

Choose a “$q$-sublattice” of $i$’s:

arithmetic progression of $i$’s where $q$ divides each $i(n + i)$.

e.g. progression $q - (n \mod q), 2q - (n \mod q), 3q - (n \mod q)$ etc.

Check smoothness of generalized congruence $i(n + i) = q$ for $i$’s in this sublattice.

e.g. check whether $i; (n + i) = q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$’s.

Rare for $i$’s to overlap.

Choose $q$, square of large prime.

Choose a “$q$-sublattice” of $i$’s:

arithmetic progression of $i$’s where $q$ divides each $i(n + i)$.

e.g. progression $q - (n \mod q), 2q - (n \mod q), 3q - (n \mod q)$ etc.

Check smoothness of generalized congruence $i(n + i) = q$ for $i$’s in this sublattice.

e.g. check whether $i; (n + i) = q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$’s.

Rare for $i$’s to overlap.
More generally, if $y \in \exp_{\mathbb{Q}} \frac{1}{2} c + o(1) \log n \log \log n$, conjectured $y$-smoothness chance is $1 = \exp_{\mathbb{Q}} r (c + 1)^2 + o(1)$.

Find enough smooth congruences by changing the range of $i$'s: replace $y^2$ with $y^{c+1} = \exp_{\mathbb{Q}} \frac{1}{2} c \log n \log \log n$.

Increasing $c$ past 1 increases number of $i$'s but reduces linear-algebra cost. So linear algebra never dominates when $y$ is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2 n$ if $i \approx y^2$.

More careful analysis:
$n + i$ doesn’t degrade, but $i$ is always smooth for $i \leq y$, only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?

Choose $q$, square of large prime. Choose a “$q$-sublattice” of $i$’s: arithmetic progression of $i$’s where $q$ divides each $i(n + i)$, e.g. progression $q - (n \mod q)$, $2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of generalized congruence $i(n + i) = q$ for $i$’s in this sublattice.
E.g. check whether $i$, $(n + i)$ are smooth for $i = q - (n \mod q)$, etc.

Try many large $q$’s.
Rare for $i$’s to overlap.
Improving smoothness chances

Smoothness chance of $i(n + i)$ degrades as $i$ grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n + i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2n$ if $i \approx y^2$.

More careful analysis:
$n + i$ doesn’t degrade, but
$i$ is always smooth for $i \leq y$,
only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?

Choose $q$, square of large prime.
Choose a “$q$-sublattice” of $i$’s:
arithmetic progression of $i$’s
where $q$ divides each $i(n + i)$.
e.g. progression $q - (n \mod q)$,
$2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of
generalized congruence $i(n + i)/q$
for $i$’s in this sublattice.
e.g. check whether $i, (n + i)/q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$’s.
Rare for $i$’s to overlap.
Improving smoothness chances

Smoothness chance of \( i(n + i) \) degrades as \( i \) grows.

For \( i \approx y^2 \) than for \( i \approx y \).

Crude analysis: \( i(n + i) \) grows.\( i \approx y \);
If \( i \approx y^2 \).

Careful analysis:
- Doesn’t degrade, but
- Always smooth for \( i \leq y \),
- Only 30% chance for \( i \approx y^2 \).

Select congruences to avoid this degradation?

Choose \( q \), square of large prime.
Choose a “\( q \)-sublattice” of \( i \)’s:
- Arithmetic progression of \( i \)’s where \( q \) divides each \( i(n + i) \).
- E.g. progression \( q - (n \mod q) \), \( 2q - (n \mod q) \), \( 3q - (n \mod q) \), etc.

Check smoothness of
generalized congruence \( i(n + i)/q \) for \( i \)’s in this sublattice.
- E.g. check whether \( i, (n+i)/q \) are smooth for \( i = q - (n \mod q) \) etc.

Try many large \( q \)’s.
Rare for \( i \)’s to overlap.

E.g. \( n = 314159265358979323 \):

Original \( Q \) sieve:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( n ) + ( i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314159265358979324</td>
</tr>
<tr>
<td>2</td>
<td>314159265358979325</td>
</tr>
<tr>
<td>3</td>
<td>314159265358979326</td>
</tr>
</tbody>
</table>

Use 997^2-sublattice,
\( i \in 802458 + 994009 \mathbb{Z} \):

<table>
<thead>
<tr>
<th>( i )</th>
<th>( n ) + ( i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>802458 316052737309</td>
</tr>
<tr>
<td>2</td>
<td>1796467 316052737310</td>
</tr>
<tr>
<td>3</td>
<td>2790476 316052737311</td>
</tr>
</tbody>
</table>
Improving smoothness chances
Smoothness chance of $i(n + i)$ degrades as $i$ grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis:
$i(n + i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2n$ if $i \approx y^2$.

More careful analysis:
$n + i$ doesn't degrade, but $i$ is always smooth for $i \leq y$,
only 30% chance for $i \approx y^2$.

Can we select congruences
to avoid this degradation?

Choose $q$, square of large prime.
Choose a “$q$-sublattice” of $i$’s:

- arithmetic progression of $i$’s where $q$ divides each $i(n + i)$.
- e.g. progression $q - (n \mod q)$,
  $2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of
generalized congruence $i(n + i)/q$
for $i$’s in this sublattice.
- e.g. check whether $i, (n + i)/q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$’s.
Rare for $i$’s to overlap.

Original $Q$ sieve:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n + i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314159265358979324</td>
</tr>
<tr>
<td>2</td>
<td>314159265358979325</td>
</tr>
<tr>
<td>3</td>
<td>314159265358979326</td>
</tr>
</tbody>
</table>

Use $997^2$-sublattice,
$i \in 802458 + 994009Z$:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$(n + i)/997^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>802458</td>
<td>316052737309</td>
</tr>
<tr>
<td>1796467</td>
<td>316052737310</td>
</tr>
<tr>
<td>2790476</td>
<td>316052737311</td>
</tr>
</tbody>
</table>
Choose \( q \), square of large prime.
Choose a “\( q \)-sublattice” of \( i \)'s:
arithmetic progression of \( i \)'s
where \( q \) divides each \( i(n + i) \).
e.g. progression \( q - (n \mod q) \),
\( 2q - (n \mod q) \), \( 3q - (n \mod q) \),
extc.

Check smoothness of
generalized congruence \( i(n + i)/q \)
for \( i \)'s in this sublattice.
e.g. check whether \( i, (n + i)/q \) are
smooth for \( i = q - (n \mod q) \) etc.

Try many large \( q \)'s.
Rare for \( i \)'s to overlap.

e.g. \( n = 314159265358979323 \):

Original \( Q \) sieve:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( n + i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314159265358979324</td>
</tr>
<tr>
<td>2</td>
<td>314159265358979325</td>
</tr>
<tr>
<td>3</td>
<td>314159265358979326</td>
</tr>
</tbody>
</table>

Use \( 997^2 \)-sublattice,
\( i \in 802458 + 994009\mathbb{Z} \):

<table>
<thead>
<tr>
<th>( i )</th>
<th>( (n + i)/997^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>802458</td>
<td>316052737309</td>
</tr>
<tr>
<td>1796467</td>
<td>316052737310</td>
</tr>
<tr>
<td>2790476</td>
<td>316052737311</td>
</tr>
</tbody>
</table>
Choose $q$, square of large prime. Choose a “$q$-sublattice” of $i$’s: arithmetic progression of $i$’s where $q$ divides each $i(n + i)$.

E.g. progression $q - (n \mod q)$, $2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of generalized congruence $i(n + i)/q$ for $i$’s in this sublattice. E.g. check whether $i, (n + i)/q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$’s. Rare for $i$’s to overlap.

E.g. $n = 314159265358979323$:

Original Q sieve:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n + i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314159265358979324</td>
</tr>
<tr>
<td>2</td>
<td>314159265358979325</td>
</tr>
<tr>
<td>3</td>
<td>314159265358979326</td>
</tr>
</tbody>
</table>

Use $997^2$-sublattice, $i \in 802458 + 994009\mathbb{Z}$:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$(n + i)/997^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>802458</td>
<td>316052737309</td>
</tr>
<tr>
<td>1796467</td>
<td>316052737310</td>
</tr>
<tr>
<td>2790476</td>
<td>316052737311</td>
</tr>
</tbody>
</table>
Choose \( q \), square of large prime.

Choose a “\( q \)-sublattice” of \( i \)’s:

- Arithmetic progression of \( i \)’s where \( q \) divides each \( i(n + i) \).

\[ q - (n \mod q), \quad 2q - (n \mod q), \quad 3q - (n \mod q), \ldots \]

Check smoothness of generalized congruence \( i(n + i) \equiv q \mod q \) in this sublattice.

Check whether \( i, \frac{(n + i)}{q} \) are smooth for \( i = q - (n \mod q) \) etc.

Try many large \( q \)’s.

Rare for \( i \)’s to overlap.

---

e.g. \( n = 314159265358979323 \):

Original \( Q \) sieve:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( n + i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314159265358979324</td>
</tr>
<tr>
<td>2</td>
<td>314159265358979325</td>
</tr>
<tr>
<td>3</td>
<td>314159265358979326</td>
</tr>
</tbody>
</table>

Use \( 997^2 \)-sublattice,

\[ i \in 802458 + 994009 \mathbb{Z} : \]

\[ \frac{(n + i)}{997^2} \]

\begin{align*}
802458 & \quad 316052737309 \\
1796467 & \quad 316052737310 \\
2790476 & \quad 316052737311 \\
\end{align*}

Crude analysis: Sublattices eliminate the growth problem.

Have practically unlimited supply of generalized congruences \( (q - (n \mod q)) \).

More careful analysis: Sublattices are even better than that!

For \( q \approx n^{1/2} \) have \( i \approx (n + i) \approx n \approx yu^2 = 2u \approx 2u \),

so smoothness chance is roughly \( \left( \frac{u}{2} \right)^{-u} \rightarrow 2^u \) times larger than before.
Choose a \( q \)-sublattice of \( i \)'s: arithmetic progression of \( i \)'s where \( q \) divides each \( i \) \((n + i)\).  
\[
\text{e.g. progression } \, q - (n \mod q),
\quad 2q - (n \mod q),
\quad 3q - (n \mod q),
\quad \text{etc.}
\]
Check smoothness of generalized congruence \( i(n + i) = q \) for \( i \)’s in this sublattice.  
\[
\text{e.g. check whether } \, i; (n + i) = q \text{ are smooth for } i = q - (n \mod q) \text{ etc.}
\]
Try many large \( q \)’s. Rare for \( i \)’s to overlap.

---

\text{e.g. } n = 314159265358979323:

\begin{align*}
\text{Original } Q \text{ sieve:} \\
\begin{array}{cc}
i & n + i \\
1 & 314159265358979324 \\
2 & 314159265358979325 \\
3 & 314159265358979326 \\
\end{array}
\end{align*}

\text{Use } 997^2\text{-sublattice, } \\
i \in 802458 + 994009\mathbb{Z}:
\begin{align*}
\begin{array}{cc}
i & (n + i)/997^2 \\
802458 & 316052737309 \\
1796467 & 316052737310 \\
2790476 & 316052737311 \\
\end{array}
\end{align*}

\text{Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences } \, (q-(n \mod q))^{n+i} \text{ between 0 and } n. \text{ More careful analysis: Sublattices are even better than that! For } q \approx n^{1/2} \text{ have } i \approx (n + i)/q \approx n, \text{ so smoothness chance is roughly } (u/2)^{-u/2}(u/2)^{-u/2} = 2^u \text{ times larger than} \, \text{et cetera.}
Choose $q$, square of large prime.

Choose a "$q$-sublattice" of $i$'s:
algebraic progression of $i$'s
where $q$ divides each $i$ ($n + i$).

e.g. progression $q - (n \mod q)$, $2q - (n \mod q)$, $3q - (n \mod q)$, etc.

Check smoothness of generalized congruence $i(n + i) = q$ for $i$'s in this sublattice.

e.g. check whether $i; (n + i) = q$ are smooth for $i = q - (n \mod q)$ etc.

Try many large $q$'s.

Rare for $i$'s to overlap.

Crude analysis: Sublattices eliminate the growth problem.

Have practically unlimited supply of generalized congruences of the form $(q - (n \mod q))^{n + q - (n \mod q)} / q$ between 0 and $n$.

More careful analysis: Sublattices are even better than that!

For $q \approx n^{1/2}$ have $i \approx (n + i) / q \approx n^{1/2} \approx y^{u/2}$ so smoothness chance is roughly $(u/2)^{-u/2} (u/2)^{-u/2} = 2^u / 2^{u^2}$ times larger than before.

\begin{align*}
\text{Original } Q \text{ sieve:} \\
\begin{array}{cc}
  i & n + i \\
  1 & 314159265358979324 \\
  2 & 314159265358979325 \\
  3 & 314159265358979326
\end{array}
\end{align*}

Use $997^2$-sublattice,
\begin{align*}
i \in 802458 + 994009 \mathbb{Z}: \\
\begin{array}{cc}
  i & (n + i) / 997^2 \\
  802458 & 316052737309 \\
  1796467 & 316052737310 \\
  2790476 & 316052737311
\end{array}
\end{align*}
e.g. \( n = 314159265358979323 \): 

Original \( \mathbb{Q} \) sieve:

\[
\begin{array}{c|c}
  i & n + i \\
  \hline
  1 & 314159265358979324 \\
  2 & 314159265358979325 \\
  3 & 314159265358979326 \\
\end{array}
\]

Use 997\(^2\)-sublattice, 
\( i \in 802458 + 994009 \mathbb{Z} \):

\[
\begin{array}{c|c}
  i & (n + i)/997^2 \\
  \hline
  802458 & 316052737309 \\
  1796467 & 316052737310 \\
  2790476 & 316052737311 \\
\end{array}
\]

Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences 
\[
(q - (n \mod q)) \frac{n + q - (n \mod q)}{q}
\]
between 0 and \( n \).

More careful analysis: Sublattices are even better than that!

For \( q \approx n^{1/2} \) have
\[
i \approx (n + i)/q \approx n^{1/2} \approx y^{u/2}
\]
so smoothness chance is roughly
\[
(u/2)^{-u/2}(u/2)^{-u/2} = 2^u / u^u,
\]
\( 2^u \) times larger than before.
Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences 

\[
(q - (n \mod q)) \frac{n+q-(n \mod q)}{q}
\]

between 0 and \( n \).

More careful analysis: Sublattices are even better than that! For \( q \approx n^{1/2} \) have 

\[
i \approx (n + i)/q \approx n^{1/2} \approx y^{u/2}
\]

so smoothness chance is roughly 

\[
(u/2)^{-u/2}(u/2)^{-u/2} = 2^u / u^u, 
\]

2^u times larger than before.

Even larger improvements from changing polynomial \( i (n + i) \).

"Quadratic sieve" (QS) uses \( i^2 - n \) with \( i \approx \sqrt{n} \); have \( i^2 - n \approx n^{1/2} + o(1) \), much smaller than \( n \).

"MPQS" improves \( o(1) \) using sublattices: \( i^2 - n = q \).

But still \( \approx n^{1/2} = 2^{u/2} \).

"Number-field sieve" (NFS) achieves \( n^{o(1)} \).
Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences
\[(q - (n \mod q)) \frac{n + q - (n \mod q)}{q}\]
between 0 and \(n\).

More careful analysis: Sublattices are even better than that! For \(q \approx n^{1/2}\) have
\[i \approx \frac{(n + i)}{q} \approx n^{1/2} \approx \sqrt{u}/2\]
so smoothness chance is roughly
\[(u/2)^{-u/2}(u/2)^{-u/2} = 2^u/u^u,\]
\(2^u\) times larger than before.

Even larger improvements from changing polynomial \(i(n + i)\).

"Quadratic sieve" (QS) uses \(i^2 - n\) with \(i \approx \sqrt{n}\) have \(i^2 - n \approx n^{1/2}\); much smaller than

"MPQS" improves \(o(1)\) using sublattices: \((i^2 - n) = q\). But still \(\approx n^{1/2}\).

"Number-field sieve" (NFS) achieves \(n^{o(1)}\).
Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences
\[(q-(n \mod q))\frac{n+q-(n \mod q)}{q}\]
between 0 and \(n\).

More careful analysis: Sublattices are even better than that! For \(q \approx n^{1/2}\) have
\[i \approx (n + i)/q \approx n^{1/2} \approx y^{u/2}\]
so smoothness chance is roughly
\[(u/2)^{-u/2}(u/2)^{-u/2} = 2^u/u^u,\]
2^u times larger than before.

Even larger improvements from changing polynomial \(i(n+i)\).

“Quadratic sieve” (QS) uses
\[i^2 - n\] with \(i \approx \sqrt{n}\); have \(i^2 - n \approx n^{1/2+o(1)}\), much smaller than \(n\).

“MPQS” improves \(o(1)\) using sublattices: \((i^2 - n)/o(1)\). But still \(\approx n^{1/2}\).

“Number-field sieve” (NFS) achieves \(n^{o(1)}\).
Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences 
\[ n + q - (n \mod q) \]
\[ q \]
between 0 and \( n \).

More careful analysis: Sublattices are even better than that! For \( q \approx n^{1/2} \) have
\[ i \approx (n + i)/q \approx n^{1/2} \approx y^{u/2} \]
so smoothness chance is roughly
\[ (u/2)^{-u/2}(u/2)^{-u/2} = 2^u / u^u , \]
\[ 2^u \] times larger than before.

Even larger improvements from changing polynomial \( i(n+i) \).

“Quadratic sieve” (QS) uses
\[ i^2 - n \] with \( i \approx \sqrt{n} \);
have \( i^2 - n \approx n^{1/2+o(1)} \), much smaller than \( n \).

“MPQS” improves \( o(1) \) using sublattices: \( (i^2 - n)/q \).
But still \( \approx n^{1/2} \).

“Number-field sieve” (NFS) achieves \( n^{o(1)} \).
Even larger improvements from changing polynomial $i(n+i)$.

“Quadratic sieve” (QS) uses $i^2 - n$ with $i \approx \sqrt{n}$; have $i^2 - n \approx n^{1/2+o(1)}$, much smaller than $n$.

“MPQS” improves $o(1)$ using sublattices: $(i^2 - n)/q$. But still $\approx n^{1/2}$.

“Number-field sieve” (NFS) achieves $n^{o(1)}$.

Generalizing beyond Q

The Q sieve is a special case of the number-field sieve.

Recall how the Q sieve factors 611:

Form a square as product of $i(i+611)j$ for several pairs $(i;j)$:

14(625) · 64(675) · 75(686) = 4410000

$\gcd\{611; 14 \cdot 64 \cdot 75 - 4410000\} = 47$. 

Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences \( q - (n \mod q) \), even between 0 and \( n \).

More careful analysis: Sublattices are even better than that! For \( q \approx n^{1/2} = 2 \) have \( i \approx (n + i) = q \approx n^{1/2} \), so smoothness chance is roughly \( 2^u/u^{1/2} \), \( 2^u/u^{1/2} \sim 2^u/u^u \), much larger than before.

Even larger improvements from changing polynomial \( i(n + i) \).

“Quadratic sieve” (QS) uses \( i^2 - n \) with \( i \approx \sqrt{n} \); have \( i^2 - n \approx n^{1/2 + o(1)} \), much smaller than \( n \).

“MPQS” improves \( o(1) \) using sublattices: \( (i^2 - n)/q \). But still \( \approx n^{1/2} \).

“Number-field sieve” (NFS) achieves \( n^{o(1)} \).

Generalizing beyond Q
The Q sieve is a special case of the number-field sieve. Recall how the Q sieve factors 611:

Form a square as product of \( i(i - j) \) for several pairs \( (i, j) \):
\[
14(625) \cdot 64(75) \cdot 75(686) = 4410000^2.
\]
\( \gcd \{ 611, 14 \cdot 64 \cdot 75 - 4410000 \} = 47. \)
Even larger improvements from changing polynomial $i(n+i)$.

“Quadratic sieve” (QS) uses $i^2 - n$ with $i \approx \sqrt{n}$; have $i^2 - n \approx n^{1/2+o(1)}$, much smaller than $n$.

“MPQS” improves $o(1)$ using sublattices: $(i^2 - n)/q$. But still $\approx n^{1/2}$.

“Number-field sieve” (NFS) achieves $n^{o(1)}$.

---

Generalizing beyond $\mathbf{Q}$

The $\mathbf{Q}$ sieve is a special case of the number-field sieve.

Recall how the $\mathbf{Q}$ sieve factors 611:

Form a square as product of $i(i + 611j)$ for several pairs $(i,j)$:

\[14(625) \cdot 64(675) \cdot 75(686) = 4410000^2.\]

$\gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\} = 47.$
Even larger improvements from changing polynomial $i(n+i)$.

"Quadratic sieve" (QS) uses $i^2 - n$ with $i \approx \sqrt{n}$;
have $i^2 - n \approx n^{1/2+o(1)}$, much smaller than $n$.

"MPQS" improves $o(1)$ using sublattices: $(i^2 - n)/q$.
But still $\approx n^{1/2}$.

"Number-field sieve" (NFS) achieves $n^{o(1)}$.

Generalizing beyond Q

The Q sieve is a special case of the number-field sieve.

Recall how the Q sieve factors 611:

Form a square as product of $i(i+611j)$ for several pairs $(i,j)$:
$14(625) \cdot 64(675) \cdot 75(686)$
$= 4410000^2$.

$\gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\}$
$= 47$. 

Even larger improvements from changing polynomial \( i(n + i) \).

"Quadratic sieve" (QS) uses \( i^2 - n \) with \( i \approx \sqrt{n} \);

\[ n = n^{1/2 + o(1)}, \]

smaller than \( n \).

"MPQS" improves \( o(1) \) using sublattices: \( (i^2 - n) = q \).

But still \( \approx n^{1/2} \).

"Number-field sieve" (NFS) achieves \( n^{o(1)} \).

Generalizing beyond Q

The Q sieve is a special case of the number-field sieve.

Recall how the Q sieve factors 611:

Form a square as product of \( i(i + 611j) \)
for several pairs \((i, j)\):

\[ 14(625) \cdot 64(675) \cdot 75(686) \]

\[ = 4410000^2. \]

Compute \( s = (-11 + 3 \cdot 25) \cdot (3 + 25) \),
\( t = 112 - 16 \cdot 25 \),
\( \gcd\{611, s - t\} = 13. \)

The Q(\(\sqrt{14}\)) sieve factors 611 as follows:

Form a square as product of \( i(i + \sqrt{14}j) \)
for several pairs \((i, j)\):

\[ (-11 + 3 \cdot 25) \cdot (3 + \sqrt{14}) \]

\[ = (112 - 16 \cdot \sqrt{14})^2. \]

Compute \( s = (-11 + 3 \cdot 25) \cdot (3 + 25) \),
\( t = 112 - 16 \cdot \sqrt{14} \),
\( \gcd\{611, s - t\} = 47. \)
Generalizing beyond \( \mathbb{Q} \)

The \( \mathbb{Q} \) sieve is a special case of the number-field sieve.

Recall how the \( \mathbb{Q} \) sieve factors 611:

Form a square as product of \((i + 25j)(i + \sqrt{14}j)\)
for several pairs \((i, j)\):

\[
14(625) \cdot 64(675) \cdot 75(686) = 4410000^2.
\]

\[
gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\} = 47.
\]

The \( \mathbb{Q}(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \((i + \sqrt{14}j)\)
for several pairs \((i, j)\):

\[
(-11 + 3 \cdot 25)(-11 + 3 \cdot \sqrt{14}) \cdot (3 + 25)(3 + \sqrt{14}) = (112 - 16\sqrt{14})^2.
\]

Compute

\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
\]
\[
t = 112 - 16 \cdot 25,
\]
\[
gcd\{611, s - t\} = 47.
\]
Even larger improvements from changing polynomial \( i(n+i) \).

"Quadratic sieve" (QS) uses \( i^2 - n \) with \( i \approx \sqrt{n} \);

have \( i^2 - n \approx n^{1.5} = 2 + o(1) \), much smaller than \( n \).

"MPQS" improves \( o(1) \) using sublattices: \( i^2 - n = q \).

But still \( \approx n^{1.5} = 2 \).

"Number-field sieve" (NFS) achieves \( n^{o(1)} \).

---

Generalizing beyond Q

The Q sieve is a special case of the number-field sieve.

Recall how the Q sieve factors 611:

Form a square as product of \( i(i + 611j) \)
for several pairs \((i,j)\):

\[
14(625) \cdot 64(675) \cdot 75(686) = 4410000^2.
\]

\[
gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\} = 47.
\]

The \( Q(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \( (i + 25j)(i + \sqrt{14}j) \)
for several pairs \((i,j)\):

\[
(-11 + 3 \cdot 25)(-11 + 3\sqrt{14}) \cdot (3 + 25)(3 + \sqrt{14}) = (112 - 16\sqrt{14})^2.
\]

Compute

\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
\]

\[
t = 112 - 16 \cdot 25,
\]

\[
gcd\{611, s - t\} = 13.
\]
Generalizing beyond \( \mathbb{Q} \)

The \( \mathbb{Q} \) sieve is a special case of the number-field sieve.

Recall how the \( \mathbb{Q} \) sieve factors 611:

Form a square as product of \( i(i + 611j) \)
for several pairs \( (i, j) \):

\[
14(625) \cdot 64(675) \cdot 75(686)
= 4410000^2.
\]

\[
gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\} = 47.
\]

The \( \mathbb{Q}(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \( (i + 25j)(i + \sqrt{14}j) \)
for several pairs \( (i, j) \):

\[
(-11 + 3 \cdot 25)(-11 + 3\sqrt{14})
\cdot (3 + 25)(3 + \sqrt{14})
= (112 - 16\sqrt{14})^2.
\]

Compute
\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
\]
\[
t = 112 - 16 \cdot 25,
\]
\[
gcd\{611, s - t\} = 13.
\]
Generalizing beyond $\mathbb{Q}$

The $\mathbb{Q}$ sieve is a special case of the number-field sieve.

Recall how the $\mathbb{Q}$ sieve factors 611:

Form a square as product of $(i + 25j)(i + \sqrt{14}j)$ for several pairs $(i,j)$:

$(-11 + 3 \cdot 25)(-11 + 3\sqrt{14}) 
\cdot (3 + 25)(3 + \sqrt{14})$ $= (112 - 16\sqrt{14})^2$.

Compute $s = (-11 + 3 \cdot 25) \cdot (3 + 25)$,

$t = 112 - 16 \cdot 25$,

$\gcd\{611, s - t\} = 13$.

Why does this work?
Answer: Have ring morphism
$\mathbb{Z}[\sqrt{14}] \to \mathbb{Z} = 611$,
$\sqrt{14} \mapsto 25$,
since $25^2 = 14$ in $\mathbb{Z} = 611$.

Apply ring morphism to square:

$(-11 + 3 \cdot 25)(-11 + 3\sqrt{14}) 
\cdot (3 + 25)(3 + \sqrt{14})$ $= (112 - 16\sqrt{14})^2$ in $\mathbb{Z} = 611$.

i.e. $s^2 = t^2$ in $\mathbb{Z} = 611$.

Unsurprising to find factor.
The \( \mathbb{Q}(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \((i + 25j)(i + \sqrt{14}j)\) for several pairs \((i, j)\):

\[
(-11 + 3 \cdot 25)(-11 + 3\sqrt{14}) \\
\cdot (3 + 25)(3 + \sqrt{14}) \\
= (112 - 16\sqrt{14})^2.
\]

Compute

\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
\]

\[
t = 112 - 16 \cdot 25,
\]

\[
\gcd\{611, s - t\} = 13.
\]

Why does this work?
Answer: Have ring morphism \( \mathbb{Z}[\sqrt{14}] \rightarrow \mathbb{Z}/611, \sqrt{14} \mapsto 25 \), since \( 25^2 = 14 \) in \( \mathbb{Z} = 611 \).

Apply ring morphism to square:

\[
(-11 + 3 \cdot 25)(-11 + 3 \cdot 25) \\
\cdot (3 + 25)(3 + 25) \\
= (112 - 16 \cdot 25)^2.
\]

i.e. \( s^2 = t^2 \) in \( \mathbb{Z}/611 \).

Unsurprising to find factor.
The \( \mathbb{Q}(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \( (i + 25j)(i + \sqrt{14}j) \)
for several pairs \((i, j)\):

\[
(-11 + 3 \cdot 25)(-11 + 3\sqrt{14})
\cdot (3 + 25)(3 + \sqrt{14})
= (112 - 16\sqrt{14})^2.
\]

Compute
\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
t = 112 - 16 \cdot 25,
gcd\{611, s - t\} = 13.
\]

Why does this work?

Answer: Have ring morphism \( \mathbb{Z}[\sqrt{14}] \to \mathbb{Z}/611, \sqrt{14} \mapsto 25 \),
since \( 25^2 = 14 \) in \( \mathbb{Z}/611 \).

Apply ring morphism to square:
\[
(-11 + 3 \cdot 25)(-11 + 3 \cdot 25)
\cdot (3 + 25)(3 + 25)
= (112 - 16 \cdot 25)^2 \text{ in } \mathbb{Z}/611.
\]
i.e. \( s^2 = t^2 \) in \( \mathbb{Z}/611 \).

Unsurprising to find factor.
The \( \mathbb{Q}(\sqrt{14}) \) sieve factors 611 as follows:

Form a square as product of \((i + 25j)(i + \sqrt{14}j)\) for several pairs \((i, j)\):

\[
(-11 + 3 \cdot 25)(-11 + 3\sqrt{14}) \\
\cdot (3 + 25)(3 + \sqrt{14}) = (112 - 16\sqrt{14})^2.
\]

Compute

\[
s = (-11 + 3 \cdot 25) \cdot (3 + 25),
\]

\[
t = 112 - 16 \cdot 25,
\]

\[
\gcd\{611, s - t\} = 13.
\]

Why does this work?
Answer: Have ring morphism \( \mathbb{Z}[\sqrt{14}] \to \mathbb{Z}/611, \sqrt{14} \mapsto 25 \), since \( 25^2 = 14 \) in \( \mathbb{Z}/611 \).

Apply ring morphism to square:

\[
(-11 + 3 \cdot 25)(-11 + 3 \cdot 25) \\
\cdot (3 + 25)(3 + 25) = (112 - 16 \cdot 25)^2 \text{ in } \mathbb{Z}/611.
\]

i.e. \( s^2 = t^2 \) in \( \mathbb{Z}/611 \).

Unsurprising to find factor.
The Q\((\sqrt{14})\) sieve factors 611 as follows:

Form a square as product of \((i + 25j)(i + \sqrt{14}j)\) for several pairs \((i,j)\):

\((\sqrt{14}\cdot 25 - 11 + 3)\cdot (3 + 25) = (112 - 16\cdot 25)^2\) in \(\mathbb{Z}/611\).

Compute \(s = (\sqrt{14}\cdot 25 - 11 + 3)\cdot (3 + 25)\), \(t = 112 - 16\cdot 25\), \(\text{gcd}\{611; s - t\} = 13\).

Why does this work?

Answer: Have ring morphism \(\mathbb{Z}[\sqrt{14}] \to \mathbb{Z}/611, \sqrt{14} \mapsto 25\), since \(25^2 = 14\) in \(\mathbb{Z}/611\).

Apply ring morphism to square:

\((-11 + 3\cdot 25)(-11 + 3\cdot 25)
\cdot (3 + 25)(3 + 25)
= (112 - 16\cdot 25)^2\) in \(\mathbb{Z}/611\).

i.e. \(s^2 = t^2\) in \(\mathbb{Z}/611\).

Unsurprising to find factor.

Generalize from \((x^2 - 14; 25)\) to \((f;m)\) with irred \(f \in \mathbb{Z}[x]\), \(m \in \mathbb{Z}\), \(f(m) \in n\mathbb{Z}\).

Write \(d = \deg f\),

\(f = f_d x^d + \cdots + f_1 x + f_0\).

Can take \(f_d = 1\) for simplicity, but larger \(f_d\) allows better parameter selection.

Pick \(r \in \mathbb{C}\), root of \(f\).

Then \(f_d r\) is a root of monic \(g = f_d - 1\cdot d f(x = f_d) \in \mathbb{Z}[x]\).

Q\((r)\leftarrow \mathbb{C}\rightarrow \mathbb{Z}[f_d r] \rightarrow \mathbb{Z}/n\mathbb{Z}\).
The Q(√14) sieve factors 611 as follows:

Form a square as product of $(i + 25j)(i + \sqrt{14}j)$ for several pairs $(i,j)$:

$(-11 + 3\cdot 25)(-11 + 3\cdot 25)
\cdot (3 + 25)(3 + 25)
= (112 - 16\cdot 25)^2$ in $\mathbb{Z}/611$.

i.e. $s^2 = t^2$ in $\mathbb{Z}/611$.

Unsurprising to find factor.

Why does this work?

Answer: Have ring morphism $\mathbb{Z}[\sqrt{14}] \to \mathbb{Z}/611$, $\sqrt{14} \mapsto 25$, since $25^2 = 14$ in $\mathbb{Z}/611$.

Apply ring morphism to square:

Can take $f_d = 1$ for simplicity, but larger $f_d$ allows better parameter selection.

Pick $r \in \mathbb{C}$, root of $f_d$.

Then $f_dr$ is a root of monic $g = f_d^{d-1}f(x = f_d^{-1})$.

Generalize from $(x^2 - 14; 25)$ to $(f,m)$ with irreducible $f \in \mathbb{Z}[x]$, $m \in \mathbb{Z}$, $f(m) \in n\mathbb{Z}$.
Why does this work?
Answer: Have ring morphism $\mathbb{Z}[\sqrt{14}] \to \mathbb{Z}/611$, $\sqrt{14} \mapsto 25$, since $25^2 = 14$ in $\mathbb{Z}/611$.

Apply ring morphism to square:
$(-11 + 3 \cdot 25)(-11 + 3 \cdot 25) \cdot (3 + 25)(3 + 25) = (112 - 16 \cdot 25)^2$ in $\mathbb{Z}/611$.

i.e. $s^2 = t^2$ in $\mathbb{Z}/611$.

Unsurprising to find factor.

Generalize from $(x^2 - 14, 25)$ to $(f, m)$ with irred $f \in \mathbb{Z}[x]$, $m \in \mathbb{Z}$, $f(m) \in n\mathbb{Z}$.

Write $d = \deg f$, $f = f_d x^d + \cdots + f_1 x^1 + f_0$.

Can take $f_d = 1$ for simplicity, but larger $f_d$ allows better parameter selection.

Pick $r \in \mathbb{C}$, root of $f$.
Then $f_d r$ is a root of monic $g = f_d^{d-1} f(x/f_d) \in \mathbb{Z}[x]$.

$\mathbb{Q}(r) \leftarrow \mathcal{O} \leftarrow \mathbb{Z}[f_d r] \xrightarrow{f_d r \mapsto f_d m}$. 
Why does this work?

Answer: Have ring morphism \(\mathbb{Z}[\sqrt{14}] \rightarrow \mathbb{Z}/611, \sqrt{14} \mapsto 25\), since \(25^2 = 14\) in \(\mathbb{Z}/611\).

Apply ring morphism to square:
\[
(-11 + 3 \cdot 25)(-11 + 3 \cdot 25) \\
\cdot (3 + 25)(3 + 25)
\]
\[
= (112 - 16 \cdot 25)^2 \text{ in } \mathbb{Z}/611.
\]
i.e. \(s^2 = t^2\) in \(\mathbb{Z}/611\).

Unsurprising to find factor.

Generalize from \((x^2 - 14, 25)\) to \((f, m)\) with irred \(f \in \mathbb{Z}[x]\), \(m \in \mathbb{Z}, f(m) \in n\mathbb{Z}\).

Write \(d = \deg f\),
\[
f = f_dx^d + \cdots + f_1x^1 + f_0x^0.
\]

Can take \(f_d = 1\) for simplicity, but larger \(f_d\) allows better parameter selection.

Pick \(r \in \mathbb{C}\), root of \(f\).

Then \(f_dr\) is a root of monic \(g = f_d^{-1}f(x/f_d) \in \mathbb{Z}[x]\).

\[
\mathbb{Q}(r) \leftarrow \mathcal{O} \leftarrow \mathbb{Z}[f_dr] \xrightarrow{f_dr \mapsto fdm} \mathbb{Z}/n
\]
Does this work? Have ring morphism \( \mathbb{Z} \to \mathbb{Z}/611, \sqrt{14} \mapsto 25, \) \( x^2 = 14 \) in \( \mathbb{Z}/611. \)

Applying ring morphism to square:
\[
(3 \cdot 25)(-11 + 3 \cdot 25)(3 + 25)(3 + 25)(-16 \cdot 25)^2
\]
\( = t^2 \) in \( \mathbb{Z}/611. \)

Unsurprising to find factor.

Generalize from \((x^2 - 14, 25)\) to \((f, m)\) with irred \( f \in \mathbb{Z}[x], m \in \mathbb{Z}, f(m) \in n\mathbb{Z}. \)

Write \( d = \deg f, \)
\[
f = f_d x^d + \cdots + f_1 x^1 + f_0 x^0.
\]

Can take \( f_d = 1 \) for simplicity, but larger \( f_d \) allows better parameter selection.

Pick \( r \in \mathbb{C}, \) root of \( f. \)
Then \( f_d r \) is a root of monic \( g = f_d^{d-1} f(x/f_d) \in \mathbb{Z}[x]. \)

\[
\mathbb{Q}(r) \leftarrow \mathcal{O} \leftarrow \mathbb{Z}[f_d r] \xrightarrow{f_d r \mapsto f_d m} \mathbb{Z}/n
\]

Could replace \( i - jx \) by higher-deg irred in \( \mathbb{Z}[x] ; \) quadratics seem fairly small for some number fields. But let’s not bother.

Say we have a square \( \mathbb{Q}(i;j) \in \mathbb{S} \) \( (i - jm)(i - jr) \) in \( \mathbb{Q}(r); \) now what?

Build square in \( \mathbb{Q}(r) \) from congruences \((i - jm)(i - jr)\) with \( i\mathbb{Z} + j\mathbb{Z} = \mathbb{Z} \) and \( j > 0. \)

\[ \prod_{(i,j) \in S} \mathbb{Q}(i;j) \in \mathbb{Q}(r); \]
Why does this work?
Answer: Have ring morphism
\[ \mathbb{Z} \xrightarrow{\sqrt{14}} \mathbb{Z} = 611, \]
\[ \sqrt{14} \mapsto 25, \]
since \( 25^2 = 14 \) in \( \mathbb{Z} = 611. \)

Apply ring morphism to square:

\[ ( -11 + 3 \cdot 25)( -11 + 3 \cdot 25) \cdot (3 + 25)(3 + 25) \]
\[ = (112 - 16 \cdot 25) \text{ in } \mathbb{Z} = 611. \]

i.e. \( s^2 = t^2 \) in \( \mathbb{Z} = 611. \)

Unsurprising to find factor.

Generalize from \((x^2 - 14, 25)\) to \((f, m)\) with irreducible \( f \in \mathbb{Z}[x], m \in \mathbb{Z}, f(m) \in n\mathbb{Z}. \)

Write \( d = \text{deg } f, \)
\[ f = f_d x^d + \cdots + f_1 x^1 + f_0 x^0. \]

Can take \( f_d = 1 \) for simplicity, but larger \( f_d \) allows better parameter selection.

Pick \( r \in \mathbb{C}, \) root of \( f. \)

Then \( f_d r \) is a root of monic \( g = f_d^{d-1} f(x/f_d) \in \mathbb{Z}[x]. \)

\[ \mathbb{Q}(r) \xrightarrow{\mathcal{O}} \mathbb{Z}[f_d r] \xrightarrow{f_d r \mapsto f_d m} \mathbb{Z}/n \]

Build square in \( \mathbb{Q}(r) \) from congruences \( (i - jm)(i - jr) \) with \( i\mathbb{Z} + j\mathbb{Z} = \mathbb{Z} \) and \( j > 0. \)

Could replace \( i - jx \) by higher-deg irreducible in \( \mathbb{Z}[x]; \) quadratics seem fairly small for some number fields.

But let's not bother.

Say we have a square \( \prod_{(i, j) \in S} (i - jm)(i - jr) \) in \( \mathbb{Q}(r); \) now what?
Why does this work?

Answer: Have ring morphism $\mathbb{Z}[\sqrt{14}] \to \mathbb{Z} = 611, \sqrt{14} \mapsto 25,$ since $25^2 = 14$ in $\mathbb{Z} = 611.$

Apply ring morphism to square:

\[
(\sqrt{11} + 3 \cdot 25)(\sqrt{11} + 3 \cdot 25) \cdot (3 + 25)(3 + 25) = (11^2 - 16 \cdot 25)^2 \text{ in } \mathbb{Z} = 611.
\]

i.e. $s^2 = t^2$ in $\mathbb{Z} = 611.$

Unsurprising to find factor.

Generalize from $(x^2 - 14; 25)$ to $(f; m)$ with irreducible $f \in \mathbb{Z}[x], m \in \mathbb{Z}, f(m) \in n\mathbb{Z}.$

Write $d = \deg f,$ $f = f_d x^d + \cdots + f_1 x + f_0 \in \mathbb{Z}[x].$

Can take $f_d = 1$ for simplicity, but larger $f_d$ allows better parameter selection.

Pick $r \in \mathbb{C},$ root of $f.$ Then $f_d r$ is a root of monic $g = f_d - 1 f_d(x = f_d) \in \mathbb{Z}[x].$

Can replace $i - jx$ by higher-deg irreducible in $\mathbb{Z}[x]$ for some number fields.

But let's not bother.

Say we have a square $\prod_{(i,j) \in S} (i - jm)(i - jr)$ in $\mathbb{Q}(r); \text{ now what?}$

Could replace $i - jm$ by $i - jr$ with number fields.

But let's not bother.

Build square in $\mathbb{Q}(r)$ from congruences $(i - jm)(i - jr)$ with $\mathbb{Z} + j\mathbb{Z} = \mathbb{Z}$ and $j > 0.$
Generalize from \((x^2 - 14, 25)\) to \((f, m)\) with irred \(f \in \mathbb{Z}[x], m \in \mathbb{Z}, f(m) \in n\mathbb{Z}\).

Write \(d = \deg f\),
\[f = f_d x^d + \cdots + f_1 x^1 + f_0 x^0.\]

Can take \(f_d = 1\) for simplicity, but larger \(f_d\) allows better parameter selection.

Pick \(r \in \mathbb{C}\), root of \(f\).
Then \(f_d r\) is a root of monic \(g = f_d^{d-1} f(x/f_d) \in \mathbb{Z}[x]\).

Build square in \(\mathbb{Q}(r)\) from congruences \((i - j m)(i - j r)\) with \(i\mathbb{Z} + j\mathbb{Z} = \mathbb{Z}\) and \(j > 0\).

Could replace \(i - j x\) by higher-deg irred in \(\mathbb{Z}[x]\); quadratics seem fairly small for some number fields. But let’s not bother.

Say we have a square \(\prod_{(i,j) \in \mathcal{S}} (i - j m)(i - j r)\) in \(\mathbb{Q}(r)\); now what?

\[\mathbb{Q}(r) \leftarrow \mathcal{O} \leftarrow \mathbb{Z}[f_d r] \xrightarrow{f_d r \mapsto f_d m} \mathbb{Z}/n\]
Build square in \( \mathbb{Q}(r) \) from congruences \((i - jm)(i - jr)\) with \(i\mathbb{Z} + j\mathbb{Z} = \mathbb{Z}\) and \(j > 0\).

Could replace \(i - jx\) by higher-deg irred in \(\mathbb{Z}[x]\); quadratics seem fairly small for some number fields. But let's not bother.

Say we have a square \(\prod_{(i,j) \in S}(i - jm)(i - jr)\) in \(\mathbb{Q}(r)\); now what?

\(\sum\) is a square in \(\mathcal{O}\), ring of integers of \(\mathbb{Q}(r)\).

Multiply by \(g'(f_d r)^2\), putting square root into \(\mathbb{Z}[f_d r]\):

\[\varphi : \mathbb{Z}[f_d r] \rightarrow \mathbb{Z} = \mathbb{Z}/n\]

Then apply the ring morphism \(\varphi\) to \(f_d r\) to \(f_d m\):

\(\varphi(r) - g'(f_d m)\) in \(\mathbb{Z}/n\) has gcd \(\gcd\{n; \varphi(r) - g'(f_d m)\}\).
Generalize from \((x^2 - 14; 25)\) to \((f; m)\) with \(f \in \mathbb{Z}[x], m \in \mathbb{Z}, f(m) \in n\mathbb{Z}\).

Write \(d = \deg f\),

\[ f = \sum_{i} f_i x^i + f_0 x^0. \]

For simplicity, let \(d = 1\) for simplicity, but larger \(d\) allows better parameter selection.

Pick \(r \in \mathbb{C}\), root of \(f\).

Then \(f_d r\) is a root of \(g = f_d - 1\) \(f(d) = f_d(r) \in \mathbb{Z}[x].\)

Build square in \(\mathbb{Q}(r)\) from congruences \((i - jm)(i - jr)\) with \(i\mathbb{Z} + j\mathbb{Z} = \mathbb{Z}\) and \(j > 0\).

Could replace \(i - jx\) by higher-deg irred in \(\mathbb{Z}[x]\); quadratics seem fairly small for some number fields. But let's not bother.

Say we have a square \(\prod_{(i,j) \in S} (i - jm)(i - jr)\) in \(\mathbb{Q}(r)\); now what?

Multiply by \(g'(f_d) f_d r\) putting square root into \(\mathbb{Z}[f_d r]\):

\[ r \leftrightarrow \mathbb{Z}/n \]

\[ \prod_{(i,j) \in S} (i - jm)(i - jr) f_d r \rightarrow f_{dm} \]

Then apply the ring morphism \(\varphi: \mathbb{Z}[f_d r] \rightarrow \mathbb{Z}/n\) taking \(f_d r\) to \(f_{dm}\). Compute \(\varphi(r) - g'(f_{dm})\) \(g'(f_{dm})^2 \prod_{(i,j) \in S} (i - jm)(i - jr)\)

In \(\mathbb{Z}/n\) have \(\varphi(r)^2 g'(f_{dm})^2 \prod_{(i,j) \in S} (i - jm)(i - jr)\).
Generalize from \((x^2 - 14; 25)\) to \((f; m)\) with \(f \in \mathbb{Z}[x]\) and \(m \in \mathbb{Z}\), \(f(m) \in n\mathbb{Z}\).

Write \(d = \deg f\),

\[
f = \cdots + f_1 x + f_0 x^0.
\]

Could replace \(i - jx\) by higher-deg irred in \(\mathbb{Z}[x]\);
quadratics seem fairly small for some number fields.
But let’s not bother.

Say we have a square
\[
\prod_{(i,j) \in S} (i - j m)(i - j r)
\]
in \(\mathbb{Q}(r)\); now what?

\[
\prod (i - j m)(i - j r) f_d^2
\]
is a square in \(\mathcal{O}\),
ring of integers of \(\mathbb{Q}(r)\).

Multiply by \(g'(f_d r)^2\),
putting square root into \(\mathbb{Z}[f_d r]\);
compute \(r\) with \(r^2 = g'(f_d r)\).

Then apply the ring morphism
\(\varphi : \mathbb{Z}[f_d r] \to \mathbb{Z}/n\) taking
\(f_d r\) to \(f_d m\). Compute \(\gcd\{\varphi(r) - g'(f_d m) \prod (i - j m) \}\)
in \(\mathbb{Z}/n\) have \(\varphi(r)^2 = g'(f_d m)^2 \prod (i - j m)^2 f_d^2\).
Build square in $\mathbb{Q}(r)$ from congruences $(i - j m)(i - j r)$ with $i \mathbb{Z} + j \mathbb{Z} = \mathbb{Z}$ and $j > 0$.

Could replace $i - j x$ by higher-deg irred in $\mathbb{Z}[x]$; quadratics seem fairly small for some number fields. But let’s not bother.

Say we have a square

$$\prod_{(i,j) \in S} (i - j m)(i - j r)$$

in $\mathbb{Q}(r)$; now what?

$\prod(i - j m)(i - j r)f_d^2$ is a square in $\mathcal{O}$, ring of integers of $\mathbb{Q}(r)$.

Multiply by $g'(f_d r)^2$, putting square root into $\mathbb{Z}[f_d r]$: compute $r$ with $r^2 = g'(f_d r)^2$. $\prod(i - j m)(i - j r)f_d^2$.

Then apply the ring morphism $\varphi : \mathbb{Z}[f_d r] \to \mathbb{Z}/n$ taking $f_d r$ to $f_d m$. Compute $\gcd\{n, \varphi(r) - g'(f_d m) \prod(i - j m)f_d\}$. In $\mathbb{Z}/n$ have $\varphi(r)^2 = g'(f_d m)^2 \prod(i - j m)^2 f_d^2$. 
Build square in \( \mathbb{Q}(r) \) from congruences \((i - jm)(i - jr)\) with \(i \mathbb{Z} + j \mathbb{Z} = \mathbb{Z}\) and \(j > 0\).

Could replace \(i - jx\) by higher-deg irred in \( \mathbb{Z}[x] \); quadratics seem fairly small for some number fields. But let's not bother.

Say we have a square \(Q(i;j) \in S(i - jm)(i - jr)\) in \(\mathbb{Q}(r)\); now what?

\[
\prod (i - jm)(i - jr)f_d^2
\]

is a square in \(O\), ring of integers of \(\mathbb{Q}(r)\).

Multiply by \(g'(f_d r)^2\), putting square root into \(\mathbb{Z}[f_d r]\):

compute \(r\) with \(r^2 = g'(f_d r)^2\).

\[
\prod (i - jm)(i - jr)f_d^2.
\]

Then apply the ring morphism \(\varphi : \mathbb{Z}[f_d r] \rightarrow \mathbb{Z}/n\) taking \(f_d r\) to \(f_d m\). Compute \(\gcd\{n, \varphi(r) - g'(f_d m) \prod (i - jm)f_d\}\).

In \(\mathbb{Z}/n\) have \(\varphi(r)^2 = g'(f_d m)^2 \prod (i - jm)^2 f_d^2\).

How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i;j)\).

Look for \(y\)-smooth congruences:

\(y\)-smooth \(i - jm\) and \(y\)-smooth \(f_d\) norm \((i - jr) = f_d j f_d^d + \cdots\).

Norm covers all roots \(r\).

Here "\(y\)-smooth" means "has no prime divisor >\(y\)."

Find enough smooth congruences.

Perform linear algebra on exponent vectors mod 2.
(r) from

\( (i - jm)(i - jr) \)

and \( j > 0 \).

\( j \times \) by

\( Z[x] \);

fairly small

fields.

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)

\( f \)
\[ \prod (i - jm)(i - jr)f_d^2 \]
is a square in \( \mathcal{O} \),
ring of integers of \( \mathbb{Q}(r) \).

Multiply by \( g'(f_d r)^2 \),
putting square root into \( \mathbb{Z}[f_d r] \):
compute \( r \) with \( r^2 = g'(f_d r)^2 \).

\[ \prod (i - jm)(i - jr)f_d^2 \]

Then apply the ring morphism
\( \varphi : \mathbb{Z}[f_d r] \to \mathbb{Z}/n \) taking
\( f_d r \) to \( f_d m \).
Compute \( \gcd \{ n, \varphi(r) - g'(f_d m) \prod (i - jm)f_d \} \).
In \( \mathbb{Z}/n \) have \( \varphi(r)^2 = g'(f_d m)^2 \prod (i - jm)^2f_d^2 \).

Find enough smooth congruences.
Perform linear algebra on exponent vectors mod 2.

How to find square product of congruences \( (i - jm)(i - jr) \)?
Start with congruences for, e.g., \( y^2 \) pairs \( (i, j) \).

Look for \( y \)-smooth congruences:
\( y \)-smooth \( i - jm \) and
\( y \)-smooth \( f_d \text{norm}(i - jr) = f_d i^d + \cdots + f_0 j^d = j^df(i/j) \).
Norm covers all \( d \) roots \( r \).
Here “\( y \)-smooth” means
“has no prime divisor > \( y \).”

Find enough smooth congruences.
\[ \prod (i - jm)(i - jr)f_d^2 \]
is a square in \( \mathcal{O} \),
ring of integers of \( \mathbb{Q}(r) \).

Multiply by \( g'(f_d r)^2 \),
putting square root into \( \mathbb{Z}[f_d r] \):
compute \( r \) with \( r^2 = g'(f_d r)^2 \).
\[ \prod (i - jm)(i - jr)f_d^2. \]

Then apply the ring morphism
\( \varphi : \mathbb{Z}[f_d r] \to \mathbb{Z}/n \) taking
\( f_d r \) to \( f_d m \). Compute \( \gcd \{ n, \varphi(r) - g'(f_d m) \prod (i - jm)f_d \} \).

In \( \mathbb{Z}/n \) have \( \varphi(r)^2 = g'(f_d m)^2 \prod (i - jm)^2 f_d^2. \)

How to find square product
of congruences \( (i - jm)(i - jr) \)?

Start with congruences for,
e.g., \( y^2 \) pairs \( (i, j) \).

Look for \( y \)-smooth congruences:
\( y \)-smooth \( i - jm \) and
\( y \)-smooth \( f_d \mathbf{norm}(i - jr) = f_d i^d + \cdots + f_0 j^d = j^d f(i/j) \).
Norm covers all \( d \) roots \( r \).

Here "\( y \)-smooth" means
"has no prime divisor > \( y \)."

Find enough smooth congruences.
Perform linear algebra on
exponent vectors mod 2.
How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i, j)\).

Look for \(y\)-smooth congruences: \(y\)-smooth \(i - jm\) and \(y\)-smooth \(f_d\text{norm}(i - jr) = f_d i^d + \cdots + f_0 j^d = j^d f(i/j)\).

Norm covers all \(d\) roots \(r\).

Here "\(y\)-smooth" means "has no prime divisor > \(y\)."

Find enough smooth congruences.

Perform linear algebra on exponent vectors mod 2.

Polynomial selection

Many \(f\)'s possible for \(n\).

How to find \(f\) that minimizes NFS time?

General strategy:

Enumerate many \(f\)'s.

For each \(f\), estimate time using information about \(f\) arithmetic,
distribution of \(j\) deg \(f\) \(f(i=j)\),
distribution of smooth numbers.
How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i, j)\).

Look for \(y\)-smooth congruences:

- \(y\)-smooth \(i - jm\) and
- \(y\)-smooth \(f_d\) \(\text{norm}(i - jr) = f_di^d + \cdots + f_0j^d = j^df(i/j)\).

Here “\(y\)-smooth” means “has no prime divisor \(> y\).”

Find enough smooth congruences.
Perform linear algebra on exponent vectors mod 2.

Polynomial selection

Many \(f\)'s possible for \(n\).

How to find \(f\) that minimizes NFS time?

General strategy:
Enumerate many \(f\)’s.

For each \(f\), estimate time using information about \(f\) arithmetic, distribution of \(j\)\(^{\text{deg} f}\), distribution of smooth numbers.
How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i, j)\).

Look for \(y\)-smooth congruences:
\(y\)-smooth \(i - jm\) and \(y\)-smooth \(\text{norm}(i - jr) = \sum_{d \geq 0} f_d i^d + \cdots + f_0 j^d = j^d f(i/j)\).

Norm covers all \(d\) roots \(r\).

Here "\(y\)-smooth" means "has no prime divisor \(> y\)."

Find enough smooth congruences.
Perform linear algebra on exponent vectors mod 2.

Polynomial selection

Many \(f\)'s possible for \(n\).

How to find \(f\) that minimizes NFS time?

General strategy:
Enumerate many \(f\)'s.
For each \(f\), estimate time using information about \(f\) arithmetic, distribution of \(j \deg f \text{norm}(i/j)\), distribution of smooth numbers.
How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i, j)\).

Look for \(y\)-smooth congruences:
- \(y\)-smooth \(i - jm\) and
- \(y\)-smooth \(f_{d^{\text{norm}}}(i - jr) = f_d i^d + \cdots + f_0 j^d = j^d f(i/j)\).

Norm covers all \(d\) roots \(r\).

Here “\(y\)-smooth” means “has no prime divisor > \(y\).”

Find enough smooth congruences.
Perform linear algebra on exponent vectors mod 2.

---

Polynomial selection

Many \(f\)’s possible for \(n\).
How to find \(f\) that minimizes NFS time?

General strategy:
Enumerate many \(f\)’s.
For each \(f\), estimate time using information about \(f\) arithmetic, distribution of \(j^{\deg f} f(i/j)\), distribution of smooth numbers.
Polynomial selection

Many $f$’s possible for $n$.

How to find $f$ that minimizes NFS time?

General strategy:

Enumerate many $f$’s.

For each $f$, estimate time using information about $f$ arithmetic, distribution of $j^\deg f f(i/j)$, distribution of smooth numbers.

Let’s restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.

Expand $n$ in base $m$:

$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.

Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.

Typically all the $f_i$’s are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)
How to find square product of congruences \((i - jm)(i - jr)\)?

Start with congruences for, e.g., \(y^2\) pairs \((i;j\) ).

Look for \(y\)-smooth congruences:
\[y\text{-smooth } \begin{aligned} i - jm & \text{ and } j - fr \text{ where } \begin{cases} j \leq y & \text{if } \deg f \leq y, \\ j = f & \text{if } \deg f = y. \end{cases} \end{aligned} \]

Here "\(y\)-smooth" means "has no prime divisor \(>y\)."

Find enough smooth congruences.
Perform linear algebra on exponent vectors mod 2.

Polynomial selection
Many \(f\)'s possible for \(n\).

How to find \(f\) that minimizes NFS time?

General strategy:
Enumerate many \(f\)'s.
For each \(f\), estimate time using information about \(f\) arithmetic, distribution of \(j^{\deg f} f(i/j)\), distribution of smooth numbers.

Let's restrict attention to \(f(x) = (x - m)(f_5x^5 + f_4x^4 + \cdots + f_0)\).

Take \(m\) near \(n^{1/6}\).
Expand \(n\) in base \(m\):
\[n = f_5m^5 + f_4m^4 + \cdots + f_0.\]

Can use negative coefficients.

Have \(f_5 \approx n^{1/6}\).
Typically all the \(f_i\)'s are on scale of \(n^{1/6}\).

(1993 Buhler Lenstra Pomerance)
Polynomial selection

Many $f$’s possible for $n$.

How to find $f$ that minimizes NFS time?

General strategy:
Enumerate many $f$’s.
For each $f$, estimate time using information about $f$ arithmetic, distribution of $j^{\deg f} f(i/j)$, distribution of smooth numbers.

Let’s restrict attention to $f(x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.
Expand $n$ in base $m$:
$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.
Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.
Typically all the $f_i$’s are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)
Polynomial selection

Many $f$’s possible for $n$.
How to find $f$ that
minimizes NFS time?

General strategy:
Enumerate many $f$’s.
For each $f$, estimate time using
information about $f$ arithmetic,
distribution of $j^{\deg f} f(i/j)$,
distribution of smooth numbers.

Let’s restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.
Expand $n$ in base $m$:
$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.
Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.
Typically all the $f_i$’s
are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)
Polynomial selection

Many f’s possible for $n$.

How to find f that minimizes NFS time?

General strategy:
Enumerate many f’s.

For each f, estimate time using information about f arithmetic,
distribution of $j_{\deg f} f(i/j)$,
distribution of smooth numbers.

Let’s restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.

Expand $n$ in base $m$:
$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.

Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.

Typically all the $f_i$’s are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)

To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^0$:

Have $f_5$, $f_4$, $f_3$, $f_2$, $f_1$ as large as $B^{0}$.

Hope that they are smaller, on scale of $B^{-1}$.

Conjecturally this happens within roughly $B^{7}$ trials.

Then $(i - jm)(f_5 i^5 + \cdots + f_0 j^5)$ is on scale of $B^{-1} R^6 n^2 = 6$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$. 

1071x52 To reduce f values by factor B:

Enumerate many possibilities
for m near B
0 :

Have f5, f4, f3, f2, f1 as large as B
.

Hope that they are smaller, on scale of B
− 1
.

Conjecturally this happens within roughly B
7
 trials.

Then (i − jm)(f5 i
5
 + \cdots + f0 j
5
) is on scale of B
− 1
R
6
n
2
 = 6
for i, j on scale of R.

Several more ways; depends on n.
Polynomial selection
Many f’s possible for n.
How to find f that minimizes NFS time?

General strategy:
Enumerate many f’s.
For each f, estimate time using information about f arithmetic, distribution of \( j \), \( \deg f \), \( f(i = j) \), distribution of smooth numbers.

Let’s restrict attention to
\( f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0) \).

Take \( m \) near \( n^{1/6} \).
Expand \( n \) in base \( m \):
\( n = f_5 m^5 + f_4 m^4 + \cdots + f_0 \).
Can use negative coefficients.

Have \( f_5 \approx n^{1/6} \).
Typically all the \( f_i \)’s are on scale of \( n^{1/6} \).

(1993 Buhler Lenstra Pomerance)

To reduce \( f \) values by factor \( B \):
Enumerate many possibilities for \( m \) near \( B^{0.25} n^{1/6} \).

Have \( f_5 \approx B^{-1.25} \), \( f_4, f_3, f_2, f_1, f_0 \) could be as large as \( B^{0.25} n^{1/6} \).
Hope that they are smaller, on scale of \( B^{-1.25} \).

Conjecturally this happens within roughly \( B^{7/25} \) trials.

Then \( (i - j m)(f_5(i) \cdots f_0(j)) \) is on scale of \( B^{-1.25} \) for \( i, j \) on scale of \( n^{1/6} \).

Several more ways; depends on \( n \).
To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^{0.25}n^{1/6}$.

Have $f_5 \approx B^{-1.25}n^{1/6}$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B^{0.25}n^{1/6}$.

Hope that they are smaller, on scale of $B^{-1.25}n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.

Then $(i - jm)(f_5i^5 + \cdots + f_0j^5)$ is on scale of $B^{-1}R^6n^{2/6}$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$.

Let’s restrict attention to $f(x) = (x - m)(f_5x^5 + f_4x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.

Expand $n$ in base $m$:

$n = f_5m^5 + f_4m^4 + \cdots + f_0$.

Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.

Typically all the $f_i$’s are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)
Let’s restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n^{1/6}$.

Expand $n$ in base $m$:
$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.

Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.

Typically all the $f_i$’s are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)

To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^{0.25} n^{1/6}$.

Have $f_5 \approx B^{-1.25} n^{1/6}$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B^{0.25} n^{1/6}$.

Hope that they are smaller, on scale of $B^{-1.25} n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.

Then $(i - j m)(f_5 i^5 + \cdots + f_0 j^5)$ is on scale of $B^{-1} R^6 n^{2/6}$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$. 
To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^{0.25} n^{1/6}$.

Have $f_5 \approx B^{-1.25} n^{1/6}$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B^{0.25} n^{1/6}$.

Hope that they are smaller, on scale of $B^{-1.25} n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.

Then $(i - jm)(f_5 i^5 + \cdots + f_0 j^5)$ is on scale of $B^{-1} R^6 n^{2/6}$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$. 

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is $L^{1.90\ldots + o(1)}$.

What are theorists' parameters?

Choose degree $d$ with $d = (\log n)^{1/3} = 3(\log \log n)^{-1} = 3 \in 1.40$. 

\(L = \exp((\log n)^{1/3})\)
Let's restrict attention to \( f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0) \).

Take \( m \) near \( n_1 = 6 \).

Expand \( n \) in base \( m \):
\[
n = f_5 m^5 + f_4 m^4 + \cdots + f_0.
\]

Can use negative coefficients.

Have \( f_5 \approx n_1 = 6 \).

Typically all the \( f_i \)'s are on scale of \( n_1 = 6 \).

(1993 Buhler Lenstra Pomerance)

To reduce \( f \) values by factor \( B \):

Enumerate many possibilities for \( m \) near \( B^{0.25} n^{1/6} \).

Have \( f_5 \approx B^{-1.25} n^{1/6} \).

\( f_4, f_3, f_2, f_1, f_0 \) could be as large as \( B^{0.25} n^{1/6} \).

Hope that they are smaller, on scale of \( B^{-1.25} n^{1/6} \).

Conjecturally this happens within roughly \( B^{7.5} \) trials.

Then \( (i - j m)(f_5 i^5 + \cdots + f_0 j^5) \) is on scale of \( B^{-1} R^6 n^{2/6} \) for \( i, j \) on scale of \( R \).

Several more ways; depends on \( n \).

Asymptotic cost exponents:

Number of bit operations in number-field sieve, with theorists' parameters, is \( L^{1.90\ldots + o(1)} \) where
\[
L = \exp((\log n)^{1/3}(\log \log n)^{2/3}).
\]

What are theorists' parameters?

Choose degree \( d \) with \( d = (\log n)^{1/3} - 1 = 3(\log \log n)^{2/3} \in 1.40\ldots + o(1) \).
Let's restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take $m$ near $n_1 = 6$.

Expand $n$ in base $m$:

$$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$$

Can use negative coefficients.

Have $f_5 \approx n_1 = 6$.

Typically all the $f_i$'s are on scale of $n_1 = 6$.

(1993 Buhler Lenstra Pomerance)

To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^{0.25} n^{1/6}$.

Have $f_5 \approx B^{-1.25} n^{1/6}$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B^{0.25} n^{1/6}$.

Hope that they are smaller, on scale of $B^{-1.25} n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.

Then $(i - jm)(f_5 i^5 + \cdots + f_0 j^5)$ is on scale of $B^{-1} R^6 n^{2/6}$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$.

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is $L^{1.90\ldots+o(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists' parameters?

Choose degree $d$ with $d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40\ldots + o(1)$.
To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B^{0.25} n^{1/6}$.

Have $f_5 \approx B^{-1.25} n^{1/6}$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B^{0.25} n^{1/6}$.

Hope that they are smaller, on scale of $B^{-1.25} n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.

Then $(i - jm)(f_5 i^5 + \cdots + f_0 j^5)$ is on scale of $B^{-1} R^6 n^{2/6}$ for $i, j$ on scale of $R$.

Several more ways; depends on $n$.

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists’ parameters, is $L^{1.90...+o(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists’ parameters?

Choose degree $d$ with $d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40\ldots + o(1)$. 

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists’ parameters, is $L^{1.90...+o(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists’ parameters?

Choose degree $d$ with $d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40\ldots + o(1)$. 

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists’ parameters, is $L^{1.90...+o(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists’ parameters?

Choose degree $d$ with $d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40\ldots + o(1)$.
To reduce \( f \) values by factor \( B \):

Enumerate many possibilities near \( B^{0.25} n^{1/6} \).

\( f_2, f_1, f_0 \) could be as large as \( B^{0.25} n^{1/6} \).

Hope that they are smaller, on scale of \( B^{-1.25} n^{1/6} \).

Conjecturally this happens within roughly \( B^{7.5} \) trials.

\((i - j m)(f_5 i^5 + \cdots + f_0 j^5)\) on scale of \( B^{-1} R^6 n^{2/6} \)

on scale of \( R \).

Several more ways; depends on \( n \).

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is \( L^{1.90\ldots + o(1)} \) where

\[ L = \exp\left((\log n)^{1/3} (\log \log n)^{2/3}\right) \]

What are theorists' parameters?

Choose degree \( d \) with

\[ d/(\log n)^{1/3} (\log \log n)^{-1/3} \in 1.40\ldots + o(1). \]

Choose integer \( m \approx n^{1/d} \).

Write \( n \) as \( m^d + f_0 \) with each \( f_k \) below \( n(1+o(1)) \).

Test smoothness of \( i - jm \) for all coprime pairs \((i;j)\) with \( 1 \leq i;j \leq L^0 : 95 \ldots + o(1) \), using primes \( \leq L^0 : 95 \ldots + o(1) \).

Conjecturally \( L^{1.90\ldots + o(1)} \) smooth values of \( i - jm \).
To reduce $f$ values by factor $B$:

Enumerate many possibilities for $m$ near $B_0$: $25^n = 6$.

Have $f_5 \approx B - 1$: $25^n = 6$.

$f_4, f_3, f_2, f_1, f_0$ could be as large as $B_0$: $25^n = 6$.

Hope that they are smaller, on scale of $B - 1$: $25^n = 6$.

Conjecturally this happens within roughly $B^{7/5}$: 5 trials.

Test smoothness of $i - jm$ for all coprime pairs $(i, j)$ with $1 \leq i, j \leq L_0$: $95 \cdots + o(1)$ pairs.

Conjecturally $L_1: 65 \cdots + o(1)$ smooth values of $i - jm$.

Choose degree $d$ with $d/(\log n)^{1/3} (\log \log n)^{-1/3} \leq 1.40 \ldots + o(1)$.

Choose $f$ with some randomness in case there are bad $f$'s.

Choose integer $m \approx n^{1/6}$; depends on $n$.

Choose integer $m \approx n^{1/6}$; depends on $n$.

Write $n$ as $m^d + f_0 + f_1 + \cdots + f_{d-1}$.

Choose integer $m \approx n^{1/6}$; depends on $n$.

Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is $L_1: 90 \cdots + o(1)$ where $L = \exp((\log \log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists' parameters?

Choose $m$ with $m^{d-1} + f^{d-1} + \cdots + f_0$ in case there are bad $f$'s.

Choose $f$ with some randomness in case there are bad $f$'s.
Choose integer $m \approx n^{1/d}$.
Write $n$ as
\[ m^d + f_{d-1}m^{d-1} + \cdots + f_1m + f_0 \]
with each $f_k$ below $n^{(1+o(1))}$.

Choose $f$ with some randomness in case there are bad $f$’s.

Test smoothness of $i - jm$ for all coprime pairs $(i,j)$ with $1 \leq i,j \leq L^{0.95...+o(1)}$, using primes $\leq L^{0.95...+o(1)}$.

$L^{1.90...+o(1)}$ pairs.
Conjecturally $L^{1.65...+o(1)}$ smooth values of $i - jm$.

Asymptotic cost exponents
Number of bit operations in number-field sieve, with theorists’ parameters, is $L^{1.90...+o(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists’ parameters?
Choose degree $d$ with
\[ d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40... + o(1). \]
Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists’ parameters, is $L^{1.90\ldots+\mathcal{o}(1)}$ where $L = \exp((\log n)^{1/3}(\log \log n)^{2/3})$.

What are theorists’ parameters?

Choose degree $d$ with $d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.40\ldots+\mathcal{o}(1)$.

Choose integer $m \approx n^{1/d}$.

Write $n$ as $m^d + f_{d-1}m^{d-1} + \cdots + f_1m + f_0$ with each $f_k$ below $n^{(1+\mathcal{o}(1))/d}$.

Choose $f$ with some randomness in case there are bad $f$’s.

Test smoothness of $i - jm$ for all coprime pairs $(i,j)$ with $1 \leq i,j \leq L^{0.95\ldots+\mathcal{o}(1)}$, using primes $\leq L^{0.95\ldots+\mathcal{o}(1)}$.

$L^{1.90\ldots+\mathcal{o}(1)}$ pairs.

Conjecturally $L^{1.65\ldots+\mathcal{o}(1)}$ smooth values of $i - jm$. 
Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is $L_1: 90^{+}o(1)$, where $L_1 = \exp((\log n)^{1/3} (\log \log n)^{-1/3})$.

What are theorists' parameters?
Choose degree $d$ with

$d = (\log n)^{1/3} (\log \log n)^{-1} = 3^{\log \log n}^{1/3} \in 1^{+}40^{+}o(1)$.

Choose integer $m \approx n^{1/d}$.
Write $n$ as

$m^d + f_{d-1}m^{d-1} + \ldots + f_1m + f_0$ with each $f_k$ below $n^{(1+o(1))/d}$.

For each coprime pairs $(i, j)$, test smoothness of $i - jm$ using primes below $L_0 = 95^{+}o(1)$.

Conjecturally $L_0: 95^{+}o(1)$ smooth congruences in case there are bad $f_k$'s.

Each $j$'s test smoothness $i - jm$.
Conjecturally $L_1: 90^{+}o(1)$ smooth values of $i - jm$.

Test smoothness of $i - jr$ and $i - j\theta$ and so on, using primes below $L_0: 95^{+}o(1)$.

For each $i - jm$, test smoothness using primes below $L_1: 97^{+}o(1)$.

Conjecturally $L_1: 97^{+}o(1)$ smooth congruences in the exponent vectors.

For each $i - jm$, test smoothness using primes below $L_0: 95^{+}o(1)$.

Use $L_0: 12^{+}o(1)$ number fields.

For each $(i, j)$ with smooth $i - jm$,
use $L_1: 12^{+}o(1)$ number fields.
Asymptotic cost exponents

Number of bit operations
in number-field sieve,
with theorists’ parameters,
is
\[ L^{\frac{1}{90} + o(1)} \]

What are theorists’ parameters?

Choose degree \( d \) with
\[ d = \left( \frac{\log n}{\log \log n} \right)^{1/3} \]

Choose integer \( m \approx n^{1/d} \).

Write \( n \) as
\[ m^d + f_{d-1}m^{d-1} + \cdots + f_1m + f_0 \]

with each \( f_k \) below \( n^{(1+o(1))/d} \).

Choose \( f \) with some randomness in case there are bad \( f \)’s.

Test smoothness of \( i - jm \)
for all coprime pairs \((i, j)\) with smooth \( i - jm \),
with \( 1 \leq i, j \leq L^{0.95...+o(1)} \),
using primes \( \leq L^{0.95...+o(1)} \).

Each \( |j^d f(i/j)| \leq L^{1.77...+o(1)} \) tests.

Conjecturally \( L^{1.90...+o(1)} \) pairs.

Conjecturally \( L^{1.65...+o(1)} \) smooth congruences.

Use \( L^{0.12...+o(1)} \) number fields.

For each \((i, j)\) with smooth \( i - j \beta \) and \( i - j \gamma \) and so on,
using primes \( \leq L^{0.95...+o(1)} \).

Each \( L^{1.12...+o(1)} \) components
in the exponent vectors.
Asymptotic cost exponents

Number of bit operations in number-field sieve, with theorists' parameters, is

\[ L_{1:90^{+o(1)}} \]

where

\[ L = \exp \left( \frac{\log n}{1 + \frac{1}{3} \log \log n - 1} \right) \].

What are theorists' parameters?

Choose degree \( d \) with

\[ d = \frac{\log n}{1 + \frac{1}{3} \log \log n - 1} \]

\[ \in 1:40^{+o(1)}. \]

Choose integer \( m \approx n^{1/d} \).

Write \( n \) as

\[ m^d + f_{d-1}m^{d-1} + \cdots + f_1m + f_0 \]

with each \( f_k \) below \( n^{(1+o(1))/d} \).

Choose \( f \) with some randomness in case there are bad \( f \)'s.

Test smoothness of \( i - jm \) for all coprime pairs \((i, j)\) with \( 1 \leq i, j \leq L_{0:95^{+o(1)}} \), using primes \( \leq L_{0:82^{+o(1)}} \).

Each \( |j^df(i/j)| \leq m_{2.86^{+o(1)}} \).

Conjecturally \( L_{0.95^{+o(1)}} \) smooth congruences.

L_{1.77^{+o(1)}} tests.

Use \( L_{0.12^{+o(1)}} \) number fields.

For each \((i, j)\) with smooth \( i - jm \), test smoothness of \( i - jr \) and \( i - j\beta \) and so on, using primes \( \leq L_{0.82^{+o(1)}} \).

Conjecturally \( L_{0.95^{+o(1)}} \) smooth congruences.

\( L_{0.95^{+o(1)}} \) components in the exponent vectors.
Choose integer $m \approx n^{1/d}$.

Write $n$ as

$$m^d + f_{d-1}m^{d-1} + \cdots + f_1m + f_0$$

with each $f_k$ below $n^{(1+o(1))/d}$.

Choose $f$ with some randomness in case there are bad $f$'s.

Test smoothness of $i - jm$ for all coprime pairs $(i,j)$ with $1 \leq i, j \leq L^{0.95...+o(1)}$, using primes $\leq L^{0.82...+o(1)}$.

$L^{1.77...+o(1)}$ tests.

Each $|j^df(i/j)| \leq m^{2.86...+o(1)}$.

Conjecturally $L^{1.65...+o(1)}$ smooth congruences.

$L^{0.95...+o(1)}$ components in the exponent vectors.

Use $L^{0.12...+o(1)}$ number fields.

For each $(i, j)$ with smooth $i - jm$, test smoothness of $i - jr$ and $i - j\beta$ and so on, using primes $\leq L^{0.82...+o(1)}$.

$L^{1.77...+o(1)}$ tests.

Each $|j^df(i/j)| \leq m^{2.86...+o(1)}$.

Conjecturally $L^{0.95...+o(1)}$ smooth congruences.

$L^{0.95...+o(1)}$ components in the exponent vectors.
Choose integer \( m \approx n^{1/d} \).

Write \( n \) as
\[
m \cdot d + f_{d-1} \cdot m^{d-1} + \cdots + f_1 m + f_0
\]
with \( f_k \) below \( n^{(1+o(1))}/d \).

If \( f \) with some randomness there are bad \( f \)'s.

Test smoothness of \( i - j \cdot m \)
for all coprime pairs \((i,j)\)
with \( 1 \leq i,j \leq L^0 : 95 \cdots + o(1) \),
using primes \( \leq L^0 : 95 \cdots + o(1) \).

Conjecturally \( L^0 : 95 \cdots + o(1) \) smooth congruences.

\( L^1.77 \cdots + o(1) \) tests.
Each \( |j^d f (i/j)| \leq m^{2.86 \cdots + o(1)} \).

Conjecturally \( L^0.95 \cdots + o(1) \) smooth congruences.

\( L^{0.95 \cdots + o(1)} \) components
in the exponent vectors.

Use \( L^{0.12 \cdots + o(1)} \) number fields.

For each \((i,j)\)
with smooth \( i - j \cdot m \),
test smoothness of \( i - j r \)
and \( i - j \beta \) and so on,
using primes \( \leq L^0 : 82 \cdots + o(1) \).

\( L^0 : 82 \cdots + o(1) \) tests.

Each \( |j^d f (i/j)| \leq m^{2.86 \cdots + o(1)} \).

Conjecturally \( L^0.95 \cdots + o(1) \) smooth congruences.

\( L^{0.95 \cdots + o(1)} \) components
in the exponent vectors.

Three sizes of numbers here:
\[ (\log n)^1 = 3 \]
\[ \log \log n \]
\[ 3 \] bits:
\[ y, i, j. \]
\[ \log n \] bits:
\[ m, i - j \cdot m, j^d f (i/j). \]

Unavoidably \( 1 = 3 \) in exponent:
usual smoothness optimization
forces \( (\log y)^2 \approx \log m \);
balancing norms with \( m \)
forces \( d \log y \approx \log m \);
and \( d \log m \approx \log n \).
Choose integer $m \approx n^{1/d}$.

Write $n$ as $m \frac{d}{d} + f \frac{d-1}{d} m - f \frac{d-1}{d} + \cdots + f_1 m + f_0$ with each $f_k$ below $n^{(1+\omega(1))/d}$.

Some randomness in case there are bad $f$’s.

Test smoothness of $i - jm$ for all coprime pairs $(i,j)$ with $1 \leq i,j \leq L_0$.

$L_0 : 95 \cdot \omega(1)$ pairs.

Conjecturally $L_1 : 65 \cdot \omega(1)$ smooth values of $i - jm$.

Use $L^{0.12\ldots+\omega(1)}$ number fields.

For each $(i,j)$ with smooth $i - jm$,
test smoothness of $i - jr$ and $i - j\beta$ and so on,
using primes $\leq L^{0.82\ldots+\omega(1)}$. $L^{1.77\ldots+\omega(1)}$ tests.

Each $|j^d f(i/j)| \leq m^{2.86\ldots+\omega(1)}$.

Conjecturally $L^{0.95\ldots+\omega(1)}$ smooth congruences.

$L^{0.95\ldots+\omega(1)}$ components in the exponent vectors.

Three sizes of numbers here:

$log n^{1/3} (log log n)^{2/3}$ bits: $y, i, j$.

$log n^{2/3} (log log n)^{1/3}$ bits: $m, i - jm, j^d f(i/j)$.

$log n$ bits: $n$.

Unavoidably $1/3$ in usual smoothness optimization forces $(log y)^2 \approx log m$;
balancing norms with $m$ forces $d log y \approx log n$ and $d log m \approx log n$. 

Three sizes of numbers here:
$(log n)^{1/3} (log log n)^{2/3}$ bits: $y, i, j$.
Choose integer \( m \approx n^1 = d \).

Write \( n \) as \( m \frac{d}{d} + f_d - 1 \frac{d}{d} - 1 + \cdots + f_1 \frac{d}{d} + f_0 \) with each \( f_k \) below \( n(1 + o(1)) = d \).

Choose \( f \) with some randomness in case there are bad \( f \)'s.

Test smoothness of \( i - jm \) for all coprime pairs \((i; j)\) with \( 1 \leq i, j \leq L_0 : 95 :: + o(1) \).

\( L_1 : 90 :: + o(1) \) pairs.

Conjecturally \( L_1 : 65 :: + o(1) \) smooth values of \( i - jm \).

Use \( L_0 : 12 :: + o(1) \) number fields.

For each \((i, j)\) with smooth \( i - jm \),

test smoothness of \( i - jr \) and \( i - j\beta \) and so on,

using primes \( \leq L_0 : 0.82... + o(1) \).

\( L_1 : 1.77... + o(1) \) tests.

Each \( |j^df(i/j)| \leq m^{2.86... + o(1)} \).

Conjecturally \( L_0 : 0.95... + o(1) \) smooth congruences.

\( L_0 : 0.95... + o(1) \) components in the exponent vectors.

Three sizes of numbers here:

\((\log n)^{1/3}(\log \log n)^{2/3}\) bits:

\( y, i, j \).

\((\log n)^{2/3}(\log \log n)^{1/3}\) bits:

\( m, i - jm, j^df(i/j) \).

\( \log n \) bits: \( n \).

Unavoidably 1/3 in exponent:

usual smoothness optimization forces \((\log y)^2 \approx \log m \);

balancing norms with \( m \) forces \( d\log y \approx \log m \);

and \( d\log m \approx \log n \).
Use $L^{0.12\ldots+o(1)}$ number fields.

For each $(i, j)$ with smooth $i - jm$,
test smoothness of $i - jr$ and $i - j\beta$ and so on,
using primes $\leq L^{0.82\ldots+o(1)}$.

$L^{1.77\ldots+o(1)}$ tests.

Each $|j^{df}(i/j)| \leq m^{2.86\ldots+o(1)}$.

Conjecturally $L^{0.95\ldots+o(1)}$ smooth congruences.

$L^{0.95\ldots+o(1)}$ components in the exponent vectors.

Three sizes of numbers here:

$(\log n)^{1/3}(\log \log n)^{2/3}$ bits:
$y, i, j$.

$(\log n)^{2/3}(\log \log n)^{1/3}$ bits:
$m, i - jm, j^{df}(i/j)$.

$\log n$ bits: $n$.

Unavoidably $1/3$ in exponent:
usual smoothness optimization forces $(\log y)^2 \approx \log m$;
balancing norms with $m$ forces $d\log y \approx \log m$;
and $d\log m \approx \log n$. 
Three sizes of numbers here:

- \((\log n)^{1/3} (\log \log n)^{2/3}\) bits: \(y, i, j\).
- \((\log n)^{2/3} (\log \log n)^{1/3}\) bits: \(m, i - jm, j^d f(i/j)\).
- \(\log n\) bits: \(n\).

Unavoidably 1/3 in exponent:
usual smoothness optimization forces \((\log y)^2 \approx \log m\);
balancing norms with \(m\) forces \(d \log y \approx \log m\);
and \(d \log m \approx \log n\).
Three sizes of numbers here:
\[(\log n)^{1/3}(\log \log n)^{2/3} \text{ bits: } y, i, j.\]

\[(\log n)^{2/3}(\log \log n)^{1/3} \text{ bits: } m, i - jm, j^{df}(i/j).\]

\[\log n \text{ bits: } n.\]

Unavoidably 1/3 in exponent:
usual smoothness optimization forces \((\log y)^2 \approx \log m;\)
balancing norms with \(m\) forces \(d\log y \approx \log m;\)
and \(d\log m \approx \log n.\]

Batch NFS

The number-field sieve used \(L^{1.90...+o(1)}\) bit operations finding smooth \(i - jm;\)
only \(L^{1.77...+o(1)}\) bit operations finding smooth \(j^{df}(i/j).\)

Many \(n\)'s can share one \(m;\)
reducing \(y\) forces \(L^{1.90...+o(1)}\) bit operations to find squares for all \(n\).

Oops, linear algebra hurts; fix by reducing \(y.\)
But still end up factoring batch in much less time than factoring each \(n\) separately.
Three sizes of numbers here:

- $(\log n)^{1/3}(\log \log n)^{2/3}$ bits: $y, i, j$.
- $(\log n)^{2/3}(\log \log n)^{1/3}$ bits: $m, i - jm, j^d f(i/j)$.
- $\log n$ bits: $n$.

Unavoidably $1/3$ in exponent: usual smoothness optimization forces $(\log y)^2 \approx \log m$; balancing norms with $m$ forces $d \log y \approx \log m$; and $d \log m \approx \log n$.

Batch NFS

The number-field sieve used $L^{1.90\ldots+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77\ldots+o(1)}$ bit operations finding smooth $j^d f(i/j)$.

Many $n$'s can share one $m$; $L^{1.90\ldots+o(1)}$ bit operations to find squares for all $n$'s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.
Three sizes of numbers here:

$(\log n)^{1/3}(\log\log n)^{2/3}$ bits: $y, i, j$.

$(\log n)^{2/3}(\log\log n)^{1/3}$ bits: $m, i - jm, j^d f(i/j)$.

$\log n$ bits: $n$.

Unavoidably $1/3$ in exponent: usual smoothness optimization forces $(\log y)^2 \approx \log m$; balancing norms with $m$ forces $d \log y \approx \log m$; and $d \log m \approx \log n$.

Batch NFS

The number-field sieve used $L^{1.90\ldots + o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77\ldots + o(1)}$ bit operations finding smooth $j^d f(i/j)$.

Many $n$’s can share one $m$; $L^{1.90\ldots + o(1)}$ bit operations to find squares for all $n$’s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.
Batch NFS

The number-field sieve used $L^{1.90...+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77...+o(1)}$ bit operations finding smooth $j^df(i/j)$.

Many $n$'s can share one $m$; $L^{1.90...+o(1)}$ bit operations to find squares for all $n$'s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.

Asymptotic batch-NFS parameters:

$d/(\log n)^{1/3} = 3$ 

$\in 1.10...+o(1)$ .

Primes $\leq L^{0.82...+o(1)}$.

Many $n$'s can share one $m$; $L^{1.64...+o(1)}$ bit operations for each target $n$.

Computer finds $L^{1.54...+o(1)}$ smooth values $i - jm$.

Computation independent of $n$ finds $L^{1.90...+o(1)}$ smooth values $i - jm$. 

$1 \leq i,j \leq L^{1.00...+o(1)}$. 

Batch NFS

The number-field sieve used $L^{1.90...+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77...+o(1)}$ bit operations finding smooth $j^df(i/j)$.

Many $n$’s can share one $m$; $L^{1.90...+o(1)}$ bit operations to find squares for all $n$’s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.
Batch NFS

The number-field sieve used $L^{1.90...+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77...+o(1)}$ bit operations finding smooth $j^{df}(i/j)$.

Many $n$’s can share one $m$; $L^{1.90...+o(1)}$ bit operations to find squares for all $n$’s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.

Asymptotic batch-NFS parameters:

$d/(\log n)^{1/3}(\log \log n)^{-1} \in 1.10... + o(1)$.

Primes $\leq L^{0.82...+o(1)}$.

$1 \leq i, j \leq L^{1.00...+o(1)}$.

Computation independent of $n$ finds $L^{1.64...+o(1)}$ smooth values $i - jm$.

$L^{1.64...+o(1)}$ operations for each target $n$. 
Batch NFS

The number-field sieve used $L^{1.90...+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77...+o(1)}$ bit operations finding smooth $j^{df}(i/j)$.

Many $n$'s can share one $m$; $L^{1.90...+o(1)}$ bit operations to find squares for all $n$'s.

Oops, linear algebra hurts; fix by reducing $y$.

But still end up factoring batch in much less time than factoring each $n$ separately.

Asymptotic batch-NFS parameters:

\[
d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.10 \ldots + o(1).
\]

Primes $\leq L^{0.82...+o(1)}$.

$1 \leq i, j \leq L^{1.00...+o(1)}$.

Computation independent of $n$ finds $L^{1.64...+o(1)}$ smooth values $i - jm$.

$L^{1.64...+o(1)}$ operations for each target $n$.
Batch NFS
The number-field sieve used $L^{1.90\ldots + o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77\ldots + o(1)}$ bit operations finding smooth $j^{df}(i/j)$.

Many $n$’s can share one $m$; $L^{1.90\ldots + o(1)}$ bit operations to find squares for all $n$’s.

Oops, linear algebra hurts; fix by reducing $y$.
But still end up factoring batch in much less time than factoring each $n$ separately.

Asymptotic batch-NFS parameters:
\[ d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.10\ldots + o(1). \]

Primes $\leq L^{0.82\ldots + o(1)}$.

1 $\leq i, j \leq L^{1.00\ldots + o(1)}$. Computation independent of $n$ finds $L^{1.64\ldots + o(1)}$ smooth values $i - jm$.
$L^{1.64\ldots + o(1)}$ operations for each target $n$. 
Batch NFS
The number-field sieve used
\( L^1 : 90 \) \( + o(1) \) bit operations
finding smooth \( i - jm \); only
\( L^1 : 77 \) \( + o(1) \) bit operations
finding smooth \( j^{df(i/j)} \).

n’s can share one \( m \);
\( L^1 : 90 \) \( + o(1) \) bit operations
finding squares for all \( n \)’s.

Oops, linear algebra hurts;
fix by reducing \( y \).

But still end up factoring
batch in much less time than
factoring each \( n \) separately.

Asymptotic batch-NFS
parameters:
\( d = (\log n)^{1/3}(\log \log n)^{-1/3} \)
\( \in 1.10 \ldots + o(1). \)

Primes \( \leq L^{0.82 \ldots + o(1)}. \)

\( 1 \leq i, j \leq L^{1.00 \ldots + o(1)}. \)

Computation independent of \( n \)
finds \( L^{1.64 \ldots + o(1)} \)
smooth values \( i - jm \).

\( L^{1.64 \ldots + o(1)} \) operations
for each target \( n \).

Batch NFS for RSA-3072
Expand \( n \) in base \( m = 2^{384} \):
\( n = n_7 m^7 + n_6 m^6 + \cdots + n_0 \)
with \( 0 \leq n_0; n_1; \ldots; n_7 < m \).

Assume irreducibility of
\( n_7 x^7 + \cdots + n_0 \).

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \):
consider pairs \( (a;b) \in \mathbb{Z} \times \mathbb{Z} \)
such that \( -H \leq a \leq H \), \( 0 < b \leq H \),
and \( \gcd\{a;b\} = 1. \)

Choose smoothness bound \( y = 2^{66} + 2^{55}. \)
Batch NFS

The number-field sieve used

\[ L_1 : 90 \cdots + o(1) \]

bit operations

finding smooth \( i - jm \); only

\[ L_1 : 77 \cdots + o(1) \]

bit operations

finding smooth \( j \).

Many \( n \)'s can share one \( m \);

\[ L_1 : 90 \cdots + o(1) \]

bit operations

to find squares for all \( n \)'s.

Oops, linear algebra hurts;

fix by reducing \( y \).

But still end up factoring

batch in much less time than

factoring each \( n \) separately.

Asymptotic batch-NFS
parameters:

\[ d = \left( \log n \right)^{1/3} \left( \log \log n \right)^{-1/3} \in 1.10 \ldots + o(1). \]

Primes \( \leq L^{0.82\ldots+o(1)} \).

\( 1 \leq i, j \leq L^{1.00\ldots+o(1)} \).

Computation independent of \( n \)

finds \( L^{1.64\ldots+o(1)} \)

smooth values \( i - jm \).

\[ L^{1.64\ldots+o(1)} \] operations

for each target \( n \).

Batch NFS for RSA-3072

Expand \( n \) in base \( m = 2^{384} \):

\[ n = n_7 m^7 + n_6 m^6 + \cdots + n_0 \]

with \( 0 \leq n_0, n_1, \ldots \leq m - 1 \).

Assume irreducibility of

\[ n_7 x^7 + n_6 x^6 + \cdots \]

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \):

consider pairs \( (a, b) \in \mathbb{Z} \times \mathbb{Z} \) such

that \( -H \leq a \leq H \), \( 0 < b \leq H \),

and \( \gcd\{a, b\} = 1 \).

Choose smoothness bound

\( y = 2^{66} + 2^{55} \).
### Asymptotic batch-NFS parameters:

\[ \frac{d}{(\log n)^{1/3}(\log \log n)^{-1/3}} \in 1.10 \ldots + o(1). \]

Primes \( \leq L^{0.82\ldots + o(1)}. \)

1 \( \leq i, j \leq L^{1.00\ldots + o(1)}. \)

Computation independent of \( n \) finds \( L^{1.64\ldots + o(1)} \) smooth values \( i - jm \).

\( L^{1.64\ldots + o(1)} \) operations for each target \( n \).

### Batch NFS for RSA-3072

Expand \( n \) in base \( m = 2^{384} \):

\[ n = n_7m^7 + n_6m^6 + \cdots + n_0 \]

with \( 0 \leq n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of \( n_7x^7 + n_6x^6 + \cdots + n_0 \).

Choose height \( H = 2^{62} + 2^{61} \) and consider pairs \( (a, b) \in \mathbb{Z} \times \mathbb{Z} \) such that \( -H \leq a \leq H, 0 < b \leq H \) and \( \gcd\{a, b\} = 1 \).

Choose smoothness bound \( y = 2^{66} + 2^{55} \).
Asymptotic batch-NFS parameters:
\[ d/(\log n)^{1/3}(\log \log n)^{-1/3} \in 1.10\ldots + o(1). \]

Primes \( \leq L^{0.82\ldots+o(1)}. \)

\[ 1 \leq i, j \leq L^{1.00\ldots+o(1)}. \]

Computation independent of \( n \) finds \( L^{1.64\ldots+o(1)} \)
smooth values \( i - jm \).

\( L^{1.64\ldots+o(1)} \) operations
for each target \( n \).

Batch NFS for RSA-3072
Expand \( n \) in base \( m = 2^{384} \):
\[ n = n_7m^7 + n_6m^6 + \cdots + n_0 \]
with \( 0 \leq n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of \( n_7x^7 + n_6x^6 + \cdots + n_0 \).

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \):
consider pairs \((a, b) \in \mathbb{Z} \times \mathbb{Z}\) such
that \( -H \leq a \leq H, 0 < b \leq H \),
and \( \gcd\{a, b\} = 1. \)

Choose smoothness bound \( y = 2^{66} + 2^{55}. \)
Asymptotic batch-NFS parameters:
\[ d = \left( \frac{\log n}{1} \right)^{1/3} \left( \log \log n \right)^{-1/3} \]
\[ \leq L^{0.82...} + o(1). \]
\[ \leq L^{1.00...} + o(1). \]

Computation independent of \( n \)
\[ 1 \leq i, j \leq L^{1.64...} + o(1) \]

Primes \( \leq L^0 : 82 \ldots \) of \( n \)
\[ \leq L^1 : 64 \ldots + o(1) \]

Batch NFS for RSA-3072
Expand \( n \) in base \( m = 2^{384} \):
\[ n = n_7 m^7 + n_6 m^6 + \cdots + n_0 \]
with \( 0 \leq n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of \( n_7 x^7 + n_6 x^6 + \cdots + n_0 \).
Choose height \( H = 2^{62} + 2^{61} + 2^{57} \):
consider pairs \( (a, b) \in \mathbb{Z} \times \mathbb{Z} \) such that
\[ -H \leq a \leq H, \ 0 < b \leq H, \]
and \( \gcd\{a, b\} = 1 \).
Choose smoothness bound \( y = 2^{66} + 2^{55} \).
Number of congruences needed \( \approx 2^y / \log y \approx 2^62 : 06 \).

Find all \( y \)-smooth \( (a - bm) c \)
\[ c = n_7 a + n_6 b + \cdots + n_0 b \]
Combine these congruences into a factorization of \( n \),
if there are enough congruences.
Asymptotic batch-NFS parameters:
\[ d = \left( \log n \right)^{-1/3} \approx 3 \left( \log \log n \right) - 1 = 3 \in 1 : 10^{51} : + o(1). \]

Primes \( \leq L_0 : 82 : + o(1) \).

1 \( \leq i, j \leq L_1 : 64 : + o(1) \).

Computation independent of \( n \) finds \( L_1 : 64 : + o(1) \) smooth values \( i - jm \).

\( L_1 : 64 : + o(1) \) operations for each target \( n \).

---

Batch NFS for RSA-3072

Expand \( n \) in base \( m = 2^{384} \):
\[ n = n_7 m^7 + n_6 m^6 + \cdots + n_0 \]
with \( 0 \leq n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of \( n_7 x^7 + n_6 x^6 + \cdots + n_0 \).

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \): consider pairs \( (a, b) \in \mathbb{Z} \times \mathbb{Z} \) such that \( -H \leq a \leq H, 0 < b \leq H \), and \( \gcd\{a, b\} = 1 \).

Choose smoothness bound \( y = 2^{66} + 2^{55} \).

Find all pairs \( (a, b) \) with \( y \)-smooth \( (a - bm)^c \) where \( c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7 \).

Combine these congruences into a factorization of \( n \), if there are enough congruences.

Number of congruences needed \( \approx 2y/\log y \approx 2^{62.06} \).

There are about \( 12H^2/\pi^2 \approx 2^{125.51} \) pairs \( (a, b) \).
Batch NFS for RSA-3072

Expand \( n \) in base \( m = 2^{384} \):
\[
n = n_7 m^7 + n_6 m^6 + \cdots + n_0
\]
with \( 0 \leq n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of \( n_7 x^7 + n_6 x^6 + \cdots + n_0 \).

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \): consider pairs \((a, b) \in \mathbb{Z} \times \mathbb{Z}\) such that \(-H \leq a \leq H, 0 < b \leq H\), and \(\gcd\{a, b\} = 1\).

Choose smoothness bound \( y = 2^{66} + 2^{55} \).

Find all pairs \((a, b)\) with \(y\)-smooth \((a - bm)c\) where
\[
c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7
\]
Combine these congruences into a factorization of \( n \), if there are enough congruences.

Number of congruences needed
\( \approx 2 y/ \log y \approx 2^{62.06} \).

There are about \( 12H^2/\pi^2 \approx 2^{125.51} \) pairs \((a, b)\).
Batch NFS for RSA-3072

Expand $n$ in base $m = 2^{384}$:

$$n = n_7 m^7 + n_6 m^6 + \cdots + n_0$$

with $0 \leq n_0, n_1, \ldots, n_7 < m$.

Assume irreducibility of

$$n_7 x^7 + n_6 x^6 + \cdots + n_0.$$ 

Choose height $H = 2^{62} + 2^{61} + 2^{57}$:
consider pairs $(a, b) \in \mathbb{Z} \times \mathbb{Z}$ such that $-H \leq a \leq H$, $0 < b \leq H$, and $\gcd\{a, b\} = 1$.

Choose smoothness bound $y = 2^{66} + 2^{55}$.

There are about

$$12H^2 / \pi^2 \approx 2^{125.51}$$
pairs $(a, b)$.

Find all pairs $(a, b)$ with $y$-smooth $(a - bm)c$ where

$$c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7.$$ 

Combine these congruences into a factorization of $n$, if there are enough congruences.

Number of congruences needed

$$\approx 2^{y/ \log y} \approx 2^{62.06}.$$
Batch NFS for RSA-3072

Expand \( n \) in base \( m = 2^{384} \):

\[ n = n_7 m^7 + n_6 m^6 + \cdots + n_0 \]

\( n_0, n_1, \ldots, n_7 < m \).

Assume irreducibility of

\[ n_7 x^7 + n_6 x^6 + \cdots + n_0. \]

Choose height \( H = 2^{62} + 2^{61} + 2^{57} \):

Consider pairs \( (a, b) \in \mathbb{Z} \times \mathbb{Z} \) such that

\[ -H \leq a \leq H, \quad 0 < b \leq H, \quad \gcd\{a, b\} = 1. \]

Smoothness bound \( 2^{55} \).

Number of congruences needed

\[ \approx 2^{\frac{y}{\log y}} \approx 2^{62.06}. \]

There are about

\[ 12H^2/\pi^2 \approx 2^{125.51} \]

pairs \( (a, b) \).

Find all pairs \( (a, b) \) with \( y \)-smooth \( (a - bm)c \) where

\[ c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7. \]

Combine these congruences into a factorization of \( n \), if there are enough congruences.

Number of congruences needed

\[ \approx 2y/\log y \approx 2^{62.06}. \]

Heuristic approximation:

\( a - bm \) has same \( y \)-smoothness chance as a uniform random integer in \([1; Hm]\), and this chance is \( u - u \) where \( u = (\log(Hm))/\log y \).

Have \( u \approx 6.707 \) and \( u - u \approx 2^{-18.42} \), so there are about \( 2^{107.09} \) pairs \( (a, b) \) such that...
There are about 
$12H^2/\pi^2 \approx 2^{125.51}$
pairs $(a, b)$.

Find all pairs $(a, b)$ with
$y$-smooth $(a - bm)c$ where 
$c = n_7a^7 + n_6a^6b + \cdots + n_0b^7$.

Combine these congruences
into a factorization of $n$,
if there are enough congruences.

Number of congruences needed 
$\approx 2y/\log y \approx 2^{62.06}$.

Heuristic approximation:
$a - bm$ has same $y$-smooth chance as a uniform random
integer in $[1, Hm]$, 
and this chance is $u - u$ where $u = (\log(Hm)) = \log y$.

Have $u \approx 6.707$ 
and $u - u \approx 2^{-18.42}$,
so there are about 
$2^{107.09}$ pairs $(a, b)$ 
such that $a - bm$
Expand $n$ in base $m = 2^{384}$:

$$n = n_7 m^7 + n_6 m^6 + \cdots + n_0$$

with $0 \leq n_0, n_1, \ldots, n_7 < m$.

Assume irreducibility of $n_7 x^7 + n_6 x^6 + \cdots + n_0$.

Choose height $H = 2^{62} + 2^{61} + 2^{57}$:

Consider pairs $(a; b) \in \mathbb{Z} \times \mathbb{Z}$ such that $-H \leq a \leq H$, $0 < b \leq H$, and $\gcd\{a; b\} = 1$.

Choose smoothness bound $y = 2^{66} + 2^{55}$.

There are about $12 H^2 / \pi^2 \approx 2^{125.51}$ pairs $(a, b)$.

Find all pairs $(a, b)$ with $y$-smooth $(a - bm)c$ where

$$c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7.$$ 

Combine these congruences into a factorization of $n$, if there are enough congruences.

Number of congruences needed

$$\approx 2y / \log y \approx 2^{62.06}.$$ 

Heuristic approximation:

$a - bm$ has same $y$-smooth chance as a uniform random integer in $[1, Hm]$, and this chance is $u^{-u}$

where $u = (\log(Hm))/\log y$.

Have $u \approx 6.707$ and $u^{-u} \approx 2^{-18.42}$, so there are about $2^{107.09}$ pairs $(a, b)$ such that $a - bm$ is smooth.
There are about \(12H^2/\pi^2 \approx 2^{125.51}\) pairs \((a, b)\).

Find all pairs \((a, b)\) with y-smooth \((a - bm)c\) where 
\[c = n_7a^7 + n_6a^6b + \cdots + n_0b^7.\]

Combine these congruences into a factorization of \(n\),
if there are enough congruences.

Number of congruences needed
\[\approx 2y/\log y \approx 2^{62.06}.\]

Heuristic approximation:
\(a - bm\) has same y-smoothness chance as a uniform random integer in \([1, Hm]\),
and this chance is \(u^{-u}\)
where \(u = (\log(Hm))/\log y\).

Have \(u \approx 6.707\)
and \(u^{-u} \approx 2^{-18.42}\),
so there are about \(2^{107.09}\) pairs \((a, b)\)
such that \(a - bm\) is smooth.
There are about $12 \approx 2^{125.51}$ pairs $(a, b)$.

Find all pairs $(a, b)$ with $y$-smooth $(a - bm)c$ where $c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7$.

Heuristic approximation:
$a - bm$ has same $y$-smoothness chance as a uniform random integer in $[1, Hm]$, and this chance is $u^{-u}$ where $u = (\log(Hm))/\log y$.

Have $u \approx 6.707$ and $u^{-u} \approx 2^{-18.42}$, so there are about $2^{107.09}$ pairs $(a, b)$ such that $a - bm$ is smooth.

Safely above $2^{62.06}$.

Heuristic approximation:
c has same $y$-smoothness chance as a uniform random integer in $[1, 8H^7 m]$, and this chance is $v^{-v}$ where $v = (\log(8H^7 m))/\log y$.

Have $v \approx 12.395$ and $v^{-v} \approx 2^{-45.01}$, so there are about $2^{62.08}$ pairs $(a, b)$ such that $a - bm$ and $c$ are both smooth.

Safely above $2^{62.06}$. 
There are about \(12 \approx 2^{125.51}\) pairs \((a; b)\).

Find all pairs \((a; b)\) with \(y\)-smooth \(a - bm\) such that
\[
c = n_7a^7 + n_6a^6b + \cdots + n_0b^7.
\]
These congruences allow us to find a factorization of \(n\), if there are enough congruences.

The number of congruences needed is
\[
\approx 2^{y = \log y} \approx 2^{62.06}
\]
Heuristic approximation:
\(c\) has same \(y\)-smoothness chance as a uniform random integer in \([1; 8H^7m]\), and this chance is \(u - v\) where \(u = (\log(Hm))/\log y\).

Have \(u \approx 6.707\) and \(u - v \approx 2^{-18.42}\), so there are about \(2^{107.09}\) pairs \((a, b)\) such that \(a - bm\) is smooth.

Safely above \(2^{62.06}\).
Heuristic approximation:
\(a - bm\) has same \(y\)-smoothness chance as a uniform random integer in \([1, Hm]\),
and this chance is \(u^{-u}\)
where \(u = (\log(Hm))/\log y\).

Have \(u \approx 6.707\)
and \(u^{-u} \approx 2^{-18.42}\),
so there are about
\(2^{107.09}\) pairs \((a, b)\)
such that \(a - bm\) is smooth.

Heuristic approximation:
c has same \(y\)-smoothness chance as a uniform random integer in \([1, 8H^7m]\),
and this chance is \(v^{-v}\)
where \(v = (\log(8H^7m))/\log y\).

Have \(v \approx 12.395\)
and \(v^{-v} \approx 2^{-45.01}\),
so there are about
\(2^{62.08}\) pairs \((a, b)\) such that \(a - bm\) and \(c\) are both smooth.

Safely above \(2^{62.06}\).
Heuristic approximation:

\[ a - bm \] has same \( y \)-smoothness chance as a uniform random integer in \([1, Hm]\),
and this chance is \( u^{-u} \)
where \( u = (\log(Hm))/\log y \).

Have \( u \approx 6.707 \)
and \( u^{-u} \approx 2^{-18.42} \),
so there are about 
\( 2^{107.09} \) pairs \((a, b)\)
such that \( a - bm \) is smooth.

Safely above \( 2^{62.06} \).
Heuristic approximation:
c has same $y$-smoothness chance as a uniform random integer in $[1, 8H^7m]$, and this chance is $v^{-v}$ where $v = (\log(8H^7m))/\log y$.

Have $v \approx 12.395$
and $v^{-v} \approx 2^{-45.01}$,
so there are about $2^{62.08}$ pairs $(a, b)$ such that $a - bm$ and $c$ are both smooth.

Safely above $2^{62.06}$.
Heuristic approximation:
c has same $y$-smoothness chance as a uniform random integer in $[1, 8H^7 m]$, and this chance is $v^{-v}$ where $v = (\log(8H^7 m)) / \log y$.

Have $v \approx 12.395$ and $v^{-v} \approx 2^{-45.01}$, so there are about $2^{62.08}$ pairs $(a, b)$ such that $a - bm$ and $c$ are both smooth.

Safely above $2^{62.06}$.

Biggest step in computation:
Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$. 
Heuristic approximation: 
$c$ has same $y$-smoothness chance as a uniform random integer in $[1, 8H^7m]$, 
and this chance is $v^{-v}$ 
where $v = (\log(8H^7m))/\log y$. 

Have $v \approx 12.395$
and $v^{-v} \approx 2^{-45.01}$,
so there are about 
$2^{62.08}$ pairs $(a, b)$ such that 
$a - bm$ and $c$ are both smooth.

Safely above $2^{62.06}$.

Biggest step in computation: 
Check $2^{125.51}$ pairs $(a, b)$
to find the $2^{107.09}$ pairs 
where $a - bm$ is smooth.

This step is independent of reused by many integers $N$. 
Heuristic approximation:
c has same $y$-smoothness chance as a uniform random integer in $[1, 8H^7 m]$, and this chance is $v^{-v}$ where $v = (\log(8H^7 m))/\log y$.

Have $v \approx 12.395$ and $v^{-v} \approx 2^{-45.01}$, so there are about $2^{62.08}$ pairs $(a, b)$ such that $a - bm$ and $c$ are both smooth.

Safely above $2^{62.06}$.

Biggest step in computation:
Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$. 
Heuristic approximation:
c has same $y$-smoothness chance as a uniform random integer in
$[1, 8H^7 m]$, and this chance is $v^{-v}$ where $v = (\log(8H^7 m))/\log y$.

Have $v \approx 12.395$
and $v^{-v} \approx 2^{-45.01}$,
so there are about $2^{62.08}$ pairs $(a, b)$ such that $a - bm$ and $c$ are both smooth.
Safely above $2^{62.06}$.

Biggest step in computation:
Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$.

Biggest step depending on $N$:
Check $2^{107.09}$ pairs $(a, b)$ to see whether $c$ is smooth.

This is much less computation! . . . or is it?
Heuristic approximation:
c has same y-smoothness chance as a uniform random integer in \([1, 8Hm]\), and this chance is \(v^{-v}\) where \(v = \left(\log(8H7m)\right)/\log y\).

Have \(v \approx 2.45.01\), and \(v^{-v} \approx 2^{-45.01}\), so there are about \(2^{62.08}\) pairs \((a, b)\) such that \(a - bm\) and \(c\) are both smooth.

Safely above \(2^{62.06}\).

Biggest step in computation:
Check \(2^{107.09}\) pairs \((a, b)\) to find the \(2^{107.09}\) pairs where \(a - bm\) is smooth.

This step is independent of \(N\), reused by many integers \(N\).

This is much less computation! ... or is it?

The \(2^{107.09}\) pairs \((a, b)\) are not consecutive, so no easy way to sieve for prime divisors of \(c\).
Heuristic approximation:
c has same \( y \)-smoothness chance as a uniform random integer in \([1; 8H^7 m]\), and this chance is \( v - v \) where 
\[
v = \frac{(\log(8H^7 m))}{\log y}.
\]
Have \( v \approx 12\) and \( v - v \approx 2 - 45 \), so there are about 
\( 2^{125.51} \) pairs \((a, b)\) such that 
\[a - bm \] and c are both smooth. 

Biggest step in computation:
Check \( 2^{125.51} \) pairs \((a, b)\) to find the \( 2^{107.09} \) pairs
where \( a - bm \) is smooth.

This step is independent of \( N \), reused by many integers \( N \).

Biggest step depending on \( N \):
Check \( 2^{107.09} \) pairs \((a, b)\) to see whether \( c \) is smooth.

This is much less computation! . . . or is it?

The \( 2^{107.09} \) pairs \((a, b)\) are not consecutive, so no easy way to sieve for prime divisors of \( c \).
Heuristic approximation: 
$c$ has same $y$-smoothness chance as a uniform random integer in $[1; 8^H m]$, and this chance is $v - v$ where $v = (\log(8^H m))^{1/2} = \log y$.

Have $v \approx 12^{395}$ and $v - v \approx 2^{45.01}$, so there are about $2^{62.08}$ pairs $(a; b)$ such that $a - bm$ and $c$ are both smooth.

Biggest step in computation:
Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$.

Biggest step depending on $N$:
Check $2^{107.09}$ pairs $(a, b)$ to see whether $c$ is smooth.

This is much less computation! … or is it?

The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$. 
Biggest step in computation:
Check $2^{125.51}$ pairs $(a, b)$
to find the $2^{107.09}$ pairs
where $a - bm$ is smooth.

This step is independent of $N$,
reused by many integers $N$.

Biggest step depending on $N$:
Check $2^{107.09}$ pairs $(a, b)$
to see whether $c$ is smooth.

This is much less
computation! . . . or is it?

The $2^{107.09}$ pairs $(a, b)$
are not consecutive,
so no easy way to sieve
for prime divisors of $c$. 
Biggest step in computation: Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$.

Biggest step depending on $N$: Check $2^{107.09}$ pairs $(a, b)$ to see whether $c$ is smooth.

This is much less computation! ... or is it?

The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
start with trial division, then Pollard rho, then Pollard $p - 1$, then ECM.
Biggest step in computation: Check $2^{125.51}$ pairs $(a, b)$ to find the $2^{107.09}$ pairs where $a - bm$ is smooth.

This step is independent of $N$, reused by many integers $N$.

Biggest step depending on $N$: Check $2^{107.09}$ pairs $(a, b)$ to see whether $c$ is smooth.

This is much less computation! . . . or is it?

The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
start with trial division, then Pollard rho, then Pollard $p - 1$, then ECM.

Most of them covered in http://facthacks.cr.yp.to/
The biggest step in computation:
125.51 pairs $(a, b)$
are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
start with trial division,
then Pollard rho,
then Pollard $p − 1$,
then ECM.

Most of them covered in
http://facthacks.cr.yp.to/

The rho method
Define $\rho_0 = 0, \rho_{k+1} = \rho_2^k + 1$.
Every prime $\leq 2^{20}$ divides
$(\rho_1 − \rho_2)(\rho_2 − \rho_4)...(\rho_{3575} − \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$
using $\approx 2^{14}$ multiplications mod $c$,
very little memory.

Compare to $\approx 2^{16}$ divisions
for trial division up to $2^{20}$. 

The 2^{107.09} pairs $(a, b)$
to find the 2^{107.09} pairs where $a − bm$ is smooth.

This step is independent of $N$, reused by many integers $N$.

The biggest step depending on $N$:
2^{107.09} pairs $(a, b)$
to see whether $c$ is smooth.
This is much less computation! … or is it?

The rho method
Define $\rho_0 = 0, \rho_{k+1} = \rho_2^k + 1$.
Every prime $\leq 2^{20}$ divides
$(\rho_1 − \rho_2)(\rho_2 − \rho_4)...(\rho_{3575} − \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$
using $\approx 2^{14}$ multiplications mod $c$,
very little memory.

Compare to $\approx 2^{16}$ divisions
for trial division up to $2^{20}$. 

The rho method
Define $\rho_0 = 0, \rho_{k+1} = \rho_2^k + 1$.
Every prime $\leq 2^{20}$ divides
$(\rho_1 − \rho_2)(\rho_2 − \rho_4)...(\rho_{3575} − \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$
using $\approx 2^{14}$ multiplications mod $c$,
very little memory.
The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
- start with trial division,
- then Pollard rho,
- then Pollard $p - 1$,
- then ECM.

Most of them covered in http://facthacks.cr.yp.to/

The rho method
Define $\rho_0 = 0$, $\rho_{k+1} = \rho_{2k} + 11$.
Every prime $\le 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_{3575} - \rho_{7150})$.
Also many larger primes.

Can compute $\gcd(c, S)$ $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$.
The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
start with trial division,
then Pollard rho,
then Pollard $p - 1$,
then ECM.

Most of them covered in http://facthacks.cr.yp.to/

The rho method
Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 1$.
Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6)\cdots(\rho_{3575} - \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$. 

\[
\rho_0 = 0, \quad \rho_{k+1} = \rho_k^2 + 1.
\]

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6)\cdots(\rho_{3575} - \rho_{7150})$.

Also many larger primes.
The $2^{107.09}$ pairs $(a, b)$ are not consecutive, so no easy way to sieve for prime divisors of $c$.

Fix: factor each number separately:
start with trial division,
then Pollard rho,
then Pollard $p - 1$,
then ECM.

Most of them covered in http://facthacks.cr.yp.to/

---

### The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})$.

Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$. 
The 2^{107} pairs \((a, b)\) are not consecutive, so no easy way to sieve for prime divisors of \(c\). Fix: factor each number separately:

- Start with trial division,
- Then Pollard rho,
- Then Pollard \(p-1\),
- Then ECM.

Most of them covered in [facthacks.cr.yp.to/](http://facthacks.cr.yp.to/)

[The rho method]

Define \(\rho_0 = 0, \rho_{k+1} = \rho_k^2 + 11\).

Every prime \(\leq 2^{20}\) divides \(S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})\).

Also many larger primes.

Can compute \(\gcd\{c, S\}\) using \(\approx 2^{14}\) multiplications mod \(c\), very little memory.

Compare to \(\approx 2^{16}\) divisions for trial division up to \(2^{20}\).

More generally: Choose \(z\).

Compute \(\gcd\{c; S\}\) where \(S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})\).

How big does \(z\) have to be for all primes \(\leq \gamma\) to divide \(S\)?

Plausible conjecture: \(\gamma_1 = 2 + o(1)\); so \(\gamma_1 = 2 + o(1)\) mults mod \(c\).

Reason: Consider first collision in \(\rho_1 \mod p; \rho_2 \mod p; \cdots\).

If \(\rho_i \mod p \neq \rho_j \mod p\) then \(\rho_k \mod p \neq \rho_{2k} \mod p\) for \(k \in (j - i) \mathbb{Z} \cap \mathbb{Z} \cap [i; \infty) \cap [j; \infty)\).
The 2
107 : 09
pairs ( a;b )
are not consecutive,
so no easy way to sieve
for prime divisors of c .
Fix: factor each number
separately:
start with trial division,
then Pollard rho,
then Pollard p − 1,
then ECM.
Most of them covered in
http://facthacks.cr.yp.to/

The rho method
Define \( \rho_0 = 0, \rho_{k+1} = \rho_k^2 + 11. \)
Every prime \( \leq 2^{20} \) divides
\( S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \)
\( \cdots (\rho_{3575} - \rho_{7150}). \)
Also many larger primes.

Can compute \( \gcd\{ c; S \} \) using
\( \approx 2^{14} \) multiplications mod c,
very little memory.

Compare to \( \approx 2^{16} \) divisions
for trial division up to \( 2^{20} \).

More generally: Choose z .
Compute \( \gcd\{ c; S \} \) where
\( S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \)
\( \cdots (\rho_{3575} - \rho_{7150}). \)
How big does z have to be
for all primes \( \leq y \) to divide
\( S \)?

Plausible conjecture:
\( y^{1/2 + o(1)} \) multiplications
so \( y^{1/2 + o(1)} \) multiplications.

Reason: Consider first collision in
\( \rho_1 \mod p, \rho_2 \mod p, \cdots \).
If \( \rho_i \mod p = \rho_j \mod p \)
then \( \rho_{k} \mod p = \rho_{2k} \mod p \)
for \( k \in (j - i)\mathbb{Z} \cap \mathbb{N} \)
The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$. Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})$. Also many larger primes.

Can compute gcd\{c, S\} using $\approx 2^{14}$ multiplications mod c, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$.

More generally: Choose $z$.
Compute gcd\{c, S\} where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod c.

Reason: Consider first collision in $\rho_1 \mod p, \rho_2 \mod p, \ldots$.
If $\rho_i \mod p = \rho_j \mod p$ then $\rho_k \mod p = \rho_{2k} \mod p$ for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [2i, \infty]$.
The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})$.

Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$.

More generally: Choose $z$.

Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod $c$.

Reason: Consider first collision in $\rho_1 \mod p, \rho_2 \mod p, \ldots$.

If $\rho_i \mod p = \rho_j \mod p$ then $\rho_k \mod p = \rho_{2k} \mod p$ for $k \in (j - i) \mathbb{Z} \cap [i, \infty] \cap [j, \infty]$. 
The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6)\cdots(\rho_{15} - \rho_{7150})$.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$,
very little memory.

Compare to $\approx 2^{16}$ divisions
for trial division up to $2^{20}$.

More generally: Choose $z$.

Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)\cdots(\rho_z - \rho_{2z})$.

How big does $z$ have to be
for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2+o(1)}$;
so $y^{1/2+o(1)}$ mults mod $c$.

Reason: Consider first collision in
$\rho_1 \mod p, \rho_2 \mod p, \ldots$
If $\rho_i \mod p = \rho_j \mod p$
then $\rho_k \mod p = \rho_{2k} \mod p$
for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p-1$ method

$S_1 = 2^{232792560} - 1$ has prime divisors $3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 53, 61, ... 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;
etc.
The rho method
Define $\rho_0 = 0$, $\rho_{k+1} = \rho_{2k} + 11$.
Every prime $\leq 2^{20}$ divides $S = \left( \rho_1 - \rho_2 \right) \left( \rho_2 - \rho_4 \right) \cdots \left( \rho_z - \rho_{2z} \right)$.

More generally: Choose $z$.
Compute $\gcd\{c, S\}$ where $S = \left( \rho_1 - \rho_2 \right) \left( \rho_2 - \rho_4 \right) \cdots \left( \rho_z - \rho_{2z} \right)$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2 + o(1)}$; so $y^{1/2 + o(1)}$ mults mod $c$.

Reason: Consider first collision in $\rho_1 \mod p, \rho_2 \mod p, \ldots$.
If $\rho_i \mod p = \rho_j \mod p$ then $\rho_k \mod p = \rho_{2k} \mod p$
for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p - 1$ method
$S_1 = 2^{232792560} - 1$ has prime divisors
\ldots$

These divisors include
70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;
etc.
The $\rho$ method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_{2^k} + 1$.

Every prime $\leq 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

Also many larger primes. Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod $c$, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to $2^{20}$.

More generally: Choose $z$.

Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2 + o(1)}$; so $y^{1/2 + o(1)}$ mults mod $c$.

Reason: Consider first collision in $\rho_1 \mod p, \rho_2 \mod p, \ldots$.

If $\rho_i \mod p = \rho_j \mod p$
then $\rho_k \mod p = \rho_{2k} \mod p$
for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p - 1$ method

$S_1 = 2^{2^{32792560}} - 1$ has prime divisors

3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 53, 61, 67, 71, 73, 79,
89, 97, 103, 109, 113, 127, 131, 137, 151, 157, 181, 191, 199 etc.

These divisors include
70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;
etc.
More generally: Choose $z$.

Compute $\gcd\{c, S\}$ where

$$S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z}).$$

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod $c$.

Reason: Consider first collision in $\rho_1 \mod p, \rho_2 \mod p, \ldots$.

If $\rho_i \mod p = \rho_j \mod p$

then $\rho_k \mod p = \rho_{2k} \mod p$

for $k \in (j - i)\mathbb{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p - 1$ method

$S_1 = 2^{232792560} - 1$ has prime divisors


These divisors include

70 of the 168 primes $\leq 10^3$;

156 of the 1229 primes $\leq 10^4$;

296 of the 9592 primes $\leq 10^5$;

470 of the 78498 primes $\leq 10^6$;

etc.
More generally: Choose $z$.
Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4) \cdots (\rho_z - \rho_{2z})$.

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

The $p - 1$ method

$S_1 = 2^{232792560} - 1$ has prime divisors


These divisors include

70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;

etc.

An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $232792560$.

Many ways for this to happen:

Why so many?

Answer: $232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$. 

$z = \lfloor \sqrt{y} \rfloor + o(1)$ 

of the $y$ primes $\leq y$.
choose \( z \).

\( S = \{ \rho_1 \cdots (\rho_z - \rho_{2z}) \} \)

How big does \( z \) have to be to divide \( S \)?

Let: \( y^{1/2+o(1)} \);

\( \mod c \).

The first collision in \( \rho_1, \rho_2, \ldots \)
\( \mod p \)
\( \rho_{2k} \mod p \)
\( [i, \infty] \cap [j, \infty] \).

These divisors include
70 of the 168 primes \( \leq 10^3 \);
156 of the 1229 primes \( \leq 10^4 \);
296 of the 9592 primes \( \leq 10^5 \);
470 of the 78498 primes \( \leq 10^6 \);

An odd prime \( p \) divides \( 2^{232792560} - 1 \) iff order of 2 in the multiplicative group \( \mod p \) divides \( s = 232792560 \).

Why so many?
Answer: \( s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \).

The \( p - 1 \) method
\( S_1 = 2^{232792560} - 1 \) has prime divisors

Many ways for this to happen:
\( 2^{232792560} \) has 960 divisors.

These divisors include
70 of the 168 primes \( \leq 10^3 \);
156 of the 1229 primes \( \leq 10^4 \);
296 of the 9592 primes \( \leq 10^5 \);
470 of the 78498 primes \( \leq 10^6 \);

etc.
More generally: Choose $z$.

Compute $\gcd\{c; S\}$ where

$$S = \left(\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdots \frac{2z-1}{2z}\right).$$

How big does $z$ have to be for all primes $\leq y$ to divide $S$?

Plausible conjecture: $y_1 = 2 + o(1)$, so $y_1 = 2 + o(1)$.

Reason: Consider first collision in $\frac{1}{2}$ mod $p; \frac{3}{4}$ mod $p; \cdots$.

If $\frac{1}{2}$ mod $p = \frac{3}{4}$ mod $p$ then $\frac{k}{2k}$ mod $p$ for $k \in (j - i) \cap [i; \infty] \cap [j; \infty]$.

### The $p-1$ method

$S_1 = 2^{232792560} - 1$ has prime divisors

3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
37, 41, 43, 53, 61, 67, 71, 73, 79,
89, 97, 103, 109, 113, 127, 131,
137, 151, 157, 181, 191, 199 etc.

These divisors include

70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;

etc.

An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen:

$232792560$ has 960 divisors.

Why so many?

Answer: $s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot \ldots$
The \( p - 1 \) method

\[ S_1 = 2^{232792560} - 1 \]

has prime divisors


These divisors include

- 70 of the 168 primes \( \leq 10^3 \);
- 156 of the 1229 primes \( \leq 10^4 \);
- 296 of the 9592 primes \( \leq 10^5 \);
- 470 of the 78498 primes \( \leq 10^6 \);
- etc.

An odd prime \( p \) divides \( 2^{232792560} - 1 \)

iff order of 2 in the multiplicative group \( \mathbb{F}_p^* \)
divides \( s = 232792560 \).

Many ways for this to happen:

\( 232792560 \) has 960 divisors.

Why so many?

Answer: \( s = 232792560 \)

\[ = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} \]
\[ = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19. \]
The \( p - 1 \) method

\[ S_1 = 2^{232792560} - 1 \] has prime
divisors

\[ 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots \]

296 of the 9592 primes \( \leq 10^5 \);
470 of the 78498 primes \( \leq 10^6 \);
etc.

An odd prime \( p \) divides \( 2^{232792560} - 1 \) iff order of 2 in the
multiplicative group \( \mathbf{F}_p^* \) divides \( s = 232792560 \).

Many ways for this to happen:

\( 2^{232792560} \) has 960 divisors.

Why so many?
Answer: \( s = 232792560 \)

\[ = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} \]

\[ = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19. \]

Can compute \( 2^{232792560} - 1 \) using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operations: 0, 1, +, −, ·.

This computation:

\[ 2^2 = 2 \cdot 1, \quad 2^{12} = 2^6 \cdot 2^6, \quad 2^{255} = 2^{110} \cdot 2^{145}, \quad 2^{3552} = 2^{1776} \cdot 2^{1776}, \quad \ldots \]

Can compute \( 2^{232792560} - 1 \) using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operations: 0, 1, +, −, ·.

This computation:

\[ 2^2 = 2 \cdot 1, \quad 2^{12} = 2^6 \cdot 2^6, \quad 2^{255} = 2^{110} \cdot 2^{145}, \quad 2^{3552} = 2^{1776} \cdot 2^{1776}, \quad \ldots \]

Can compute \( 2^{232792560} - 1 \) using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operations: 0, 1, +, −, ·.

This computation:
The $p-1$ method

$s_1 = 2^{232792560} - 1$ has prime
divisors

3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
37, 41, 43, 53, 61, ... 10

These divisors include
70 of the 168 primes $\leq 10^3$;
156 of the 1229 primes $\leq 10^4$;
296 of the 9592 primes $\leq 10^5$;
470 of the 78498 primes $\leq 10^6$;

An odd prime $p$
divides $2^{232792560} - 1$
iff order of 2 in the
multiplicative group $\mathbf{F}_p^*$
divides $s = 232792560$.

Many ways for this to happen:
232792560 has 960 divisors.

Why so many?
Answer: $s = 232792560$

$= \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\}$

$= 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$
using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operation: 0, 1, +, − , ·...

This computation:

$2^2 = 2 \cdot 2$; $2^3 = 2^2 \cdot 2$;
$2^{12} = 2^6 \cdot 2^6$; $2^{13} = 2^6 \cdot 2^6 \cdot 2$;
$2^{55}$; $2^{110}$; $2^{111}$; $2^{222}$;
$2^{3552}$; $2^{7104}$; $2^{14208}$;
$2^{56834}$; $2^{113668}$; $2^{227336}$;
$2^{909345}$; $2^{1818690}$; $2^{3637383}$;
$2^{7274766}$; $2^{14549535}$; $2^{29099070}$;
$2^{116396280}$; $2^{232792560}$.
The $p-1$ method $S_1 = 2^{232792560} - 1$ has prime divisors $3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 53, 61, 67, 71, 73, 79, 89, 97, 103, 109, 113, 127, 131, 137, 151, 157, 181, 191, 199$ etc.

These divisors include $70$ of the $168$ primes $\leq 10^3$;
$156$ of the $1229$ primes $\leq 10^4$;
$296$ of the $9592$ primes $\leq 10^5$;
$470$ of the $78498$ primes $\leq 10^6$;
etc.

An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen:
$232792560$ has $960$ divisors.

Why so many?
Answer: $s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$ using $41$ ring operations.
(Side note: $41$ is not minimal.)

Ring operation: $0, 1, +, -, \cdot$ etc.

This computation: $1; 2 = 1 + 1$; $2^2 = 2 \cdot 2; 2^3 = 2^2 \cdot 2; 2^6 = 2^3 \cdot 2$;
$2^{12} = 2^6 \cdot 2^6; 2^{13} = 2^{12} \cdot 2; 2^{26} = 2^{13} \cdot 2$;
$2^{55}; 2^{110}; 2^{111}; 2^{222}; 2^{444}; 2^{888}$;
$2^{1776}; 2^{3552}; 2^{7104}; 2^{14208}; 2^{28416}; 2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}$;
$2^{909345}; 2^{1818690}; 2^{3637383}; 2^{7274766}; 2^{7274767}; 2^{14549535}; 2^{29099070}; 2^{58198140}$;
$2^{116396280}; 2^{232792560}; 2^{232792560} - 1$. 
An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen: $232792560$ has 960 divisors.

Why so many?
Answer: $s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$ using 41 ring operations. (Side note: 41 is not minimal.)

Ring operation: 0, 1, +, −, ·.

This computation: $1; 2 = 1 + 1; 2^2 = 2 \cdot 2; 2^3 = 2^2 \cdot 2; 2^6 = 2^3 \cdot 2^3; 2^{12} = 2^6 \cdot 2^6; 2^{13} = 2^{12} \cdot 2; 2^{26} = 2^{27} \cdot 2^{54}; 2^{55} = 2^{110}; 2^{111} = 2^{222}; 2^{444} = 2^{888}; 2^{1776}; 2^{3552} = 2^{7104}; 2^{14208}; 2^{28416}; 2^{28417}; 2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}; 2^{909344}; 2^{909345}; 2^{1818690}; 2^{1818691} = 2^{3637382}; 2^{3637383}; 2^{7274766}; 2^{7274767}; 2^{14549534}; 2^{14549535}; 2^{29099070}; 2^{58198140}; 2^{116396280} = 2^{232792560}; 2^{232792560} - 1$. 
An odd prime $p$ divides $2^{232792560} - 1$ if and only if the order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen: $s = 232792560$ has 960 divisors.

Why so many?

Answer: $s = 232792560 = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 20\} = 2^{14} \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$ using 41 ring operations.

(Side note: 41 is not minimal.)

Ring operation: 0, 1, +, −, · ...

This computation: $1; 2 = 1 + 1; 2^2 = 2 \cdot 2; 2^3 = 2^2 \cdot 2; 2^6 = 2^3 \cdot 2^3; 2^{12} = 2^6 \cdot 2^6; 2^{13} = 2^{12} \cdot 2; 2^{26}; 2^{27}; 2^{54}; 2^{55}; 2^{110}; 2^{111}; 2^{222}; 2^{444}; 2^{888}; 2^{1776}; 2^{3552}; 2^{7104}; 2^{14208}; 2^{28416}; 2^{28417}; 2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}; 2^{909344}; 2^{909345}; 2^{1818690}; 2^{1818691}; 2^{3637382}; 2^{3637383}; 2^{7274766}; 2^{7274767}; 2^{14549534}; 2^{14549535}; 2^{29099070}; 2^{58198140}; 2^{116396280}; 2^{232792560}; 2^{232792560} - 1$.

Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z} = c$.

Notation: $a \mod b = a - b \cdot \lfloor a/b \rfloor$.

e.g. $c = 8597231219$:

$2^27 \mod c = 134217728; 2^{54} \mod c = 134217728; 2^{55} \mod c = 1871327032; 2^{110} \mod c = 1871327032; 2^{222} \mod c = 1458876811; \ldots ; 2^{232792560} - 1 \mod c = 5626089344$. 

232792560
An odd prime $p$ divides $2^{232792560} - 1$ iff the order of 2 in the multiplicative group $\mathbb{F}_p^*$ divides $s = 232792560$.

Many ways for this to happen: $232792560$ has 960 divisors.

Why so many? Answer: $s = 232792560 = \text{lcm} \{ 1; 2; 3; 4; 5; \ldots; 20 \} = 2^{14} \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$ using 41 ring operations. (Side note: 41 is not minimal.)

Ring operation: 0, 1, +, −, ·.

This computation:
1; 2 = 1 + 1;
$2^2 = 2 \cdot 2$; $2^3 = 2^2 \cdot 2$; $2^6 = 2^3 \cdot 2^3$;
$2^{12} = 2^6 \cdot 2^6$; $2^{13} = 2^{12} \cdot 2$; $2^{26}$; $2^{27}$; $2^{54}$;
$2^{55}$; $2^{110}$; $2^{111}$; $2^{222}$; $2^{444}$; $2^{888}$; $2^{1776}$;
$2^{3552}$; $2^{7104}$; $2^{14208}$; $2^{28416}$; $2^{28417}$;
$2^{56834}$; $2^{113668}$; $2^{227336}$; $2^{454672}$; $2^{909344}$;
$2^{909345}$; $2^{1818690}$; $2^{1818691}$; $2^{3637382}$;
$2^{3637383}$; $2^{7274766}$; $2^{7274767}$; $2^{14549534}$;
$2^{14549535}$; $2^{29099070}$; $2^{58198140}$;
$2^{116396280}$; $2^{232792560}$; $2^{232792560} - 1$.

Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z} = c$.

Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

e.g. $c = 8597231219$:
$2^{27} \mod c = 134217728$;
$2^{54} \mod c = 134217728$;
$2^{55} \mod c = 935663516$;
$2^{110} \mod c = 1871327032$;
$2^{232792560} - 1 \mod c = 5626089344$. 
An odd prime $p$ divides $2^{232792560} - 1$ iff order of 2 in the multiplicative group $F^*_p$ divides $s = 232792560$.

Many ways for this to happen: $232792560$ has 960 divisors. Why so many? Answer: $s = 232792560 = \text{lcm} \{1; 2; 3; 4; 5; \ldots; 20\} = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

Can compute $2^{232792560} - 1$ using 41 ring operations. (Side note: 41 is not minimal.)

Ring operation: 0, 1, +, −, ·...

This computation: $1; 2 = 1 + 1;
2^2 = 2 \cdot 2; 2^3 = 2^2 \cdot 2; 2^6 = 2^3 \cdot 2^3;
2^{12} = 2^6 \cdot 2^6; 2^{13} = 2^{12} \cdot 2; 2^{26}; 2^{27}; 2^{54};
2^{55}; 2^{110}; 2^{111}; 2^{222}; 2^{444}; 2^{888}; 2^{1776};
2^{3552}; 2^{7104}; 2^{14208}; 2^{28416}; 2^{28417};
2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}; 2^{909344};
2^{909345}; 2^{1818690}; 2^{1818691}; 2^{3637382};
2^{3637383}; 2^{7274766}; 2^{7274767}; 2^{14549534};
2^{14549535}; 2^{29099070}; 2^{58198140};
2^{116396280}; 2^{232792560}; 2^{232792560} - 1.$

Given positive integer $n$, can compute $2^{232792560} - 1$ mod $c$ using 41 operations in $\mathbb{Z}/c$. Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

E.g. $c = 8597231219$: \ldots 
$2^{27} \mod c = 134217728$;
$2^{54} \mod c = 134217728^2 \mod c = 935663516$;
$2^{55} \mod c = 1871327032$;
$2^{110} \mod c = 1871327032^2 \mod c = 1458876811$; \ldots 
$2^{232792560} - 1 \mod c = 5626089344$. 

19.
Can compute $2^{232792560} - 1$ using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operation: $0, 1, +, -, \cdot$...

This computation: $1; 2 = 1 + 1; 2^2 = 2 \cdot 2; 2^3 = 2^2 \cdot 2; 2^6 = 2^3 \cdot 2^3; 2^{12} = 2^6 \cdot 2^6; 2^{13} = 2^{12} \cdot 2; 2^{26}; 2^{27}; 2^{54}; 2^{55}; 2^{110}; 2^{111}; 2^{222}; 2^{444}; 2^{888}; 2^{1776}; 2^{3552}; 2^{7104}; 2^{14208}; 2^{28416}; 2^{28417}; 2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}; 2^{909344}; 2^{909345}; 2^{1818690}; 2^{1818691}; 2^{3637382}; 2^{3637383}; 2^{7274766}; 2^{7274767}; 2^{14549534}; 2^{14549535}; 2^{29099070}; 2^{58198140}; 2^{116396280}; 2^{232792560}; 2^{232792560} - 1.$

Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z}/c$.
Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

e.g. $c = 8597231219$: $\ldots$

$2^{27} \mod c = 134217728; 2^{54} \mod c = 134217728^2 \mod n = 935663516; 2^{55} \mod c = 1871327032; 2^{110} \mod c = 1871327032^2 \mod c = 1458876811; \ldots; 2^{232792560} - 1 \mod c = 5626089344.$
Can compute $2^{232792560} - 1$ using 41 ring operations.
(Side note: 41 is not minimal.)

Ring operation: 0, 1, +, −, ·.

This computation: $1; 2 = 1 + 1$;
$2^2 = 2 \cdot 2$; $2^3 = 2^2 \cdot 2$; $2^6 = 2^3 \cdot 2^3$;
$2^{12} = 2^6 \cdot 2^6$; $2^{13} = 2^{12} \cdot 2$; $2^{26}$; $2^{27}$; $2^{54}$;
$2^{55}$; $2^{110}$; $2^{111}$; $2^{222}$; $2^{444}$; $2^{888}$; $2^{1776}$;
$2^{3552}$; $2^{7104}$; $2^{14208}$; $2^{28416}$; $2^{28417}$;
$2^{56834}$; $2^{113668}$; $2^{227336}$; $2^{454672}$; $2^{909344}$;
$2^{909345}$; $2^{1818690}$; $2^{1818691}$; $2^{3637382}$;
$2^{3637383}$; $2^{7274766}$; $2^{7274767}$; $2^{14549534}$;
$2^{14549535}$; $2^{29099070}$; $2^{58198140}$;
$2^{116396280}$; $2^{232792560}$; $2^{232792560} - 1$.

Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$
using 41 operations in $\mathbb{Z}/c$.
Notation: $a \mod b = a - b \left\lfloor a/b \right\rfloor$.

e.g. $c = 8597231219$: 
$2^{27} \mod c = 134217728$;
$2^{54} \mod c = 134217728^2 \mod n$
$= 935663516$;
$2^{55} \mod c = 1871327032$;
$2^{110} \mod c = 1871327032^2 \mod c$
$= 1458876811$; 
$2^{232792560} - 1 \mod c = 5626089344$.

Easy extra computation (Euclid):
gcd\{5626089344, c\} = 991.
Given positive integer \( n \),
can compute \( 2^{2^{32792560}} - 1 \) mod \( c \) using 41 operations in \( \mathbb{Z}/c \).

Notation: \( a \mod b = a - b \left\lfloor a/b \right\rfloor \).

e.g. \( c = 8597231219 \): \( \cdots \)
\[

gcd \{ 5626089344, c \} = 991.
\]

This \( p - 1 \) method (1974 Pollard)
quickly factored \( c = 8597231219 \).
Main work: 27 squarings mod \( c \).

Could instead have checked \( c \)'s divisibility by \( 2; 3; 5; \cdots \).
The 167th trial division
would have found divisor 991.

Not clear which method is better.
Dividing by small \( p \) is faster than squaring mod \( c \).
The \( p - 1 \) method finds
only 70 of the primes \( \leq 1000 \);
trial division finds all 168 primes.
Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z}/c$.
Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

e.g. $c = 8597231219$: 

$2^{27} \mod c = 134217728$; 

$2^{54} \mod c = 134217728^2 \mod n = 935663516$; 

$2^{55} \mod c = 1871327032$; 

$2^{110} \mod c = 1871327032^2 \mod c = 1458876811$; 

$2^{232792560} - 1 \mod c = 5626089344$.

Easy extra computation (Euclid): $\gcd\{5626089344, c\} = 991$. 

This $p - 1$ method quickly factored $c = 8597231219$. 
Main work: 27 squarings mod $c$. 
Could instead have checked $c$'s divisibility by $2, 3, 5, \ldots$. 
The 167th trial division would have found divisor 991. 
Not clear which method is better. 
Dividing by small $p$ is faster than squaring mod $c$. 
The $p - 1$ method finds only 70 of the primes $\leq 1000$; 
trial division finds all 168 primes.
Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z}/c$.

Notation: $a \mod b = a - b\lfloor a/b \rfloor$.

e.g. $c = 8597231219$: ...

$2^{27} \mod c = 134217728$;

$2^{54} \mod c = 134217728^2 \mod n = 935663516$;

$2^{55} \mod c = 1871327032$;

$2^{110} \mod c = 1871327032^2 \mod c = 1458876811$; ...

$2^{232792560} - 1 \mod c = 5626089344$.

Easy extra computation (Euclid):

$\gcd\{5626089344, c\} = 991$.

This $p-1$ method (1974 Pollard) quickly factored $c = 8597231219$.

Main work: 27 squarings mod $c$.

Could instead have checked $c$'s divisibility by 2, 3, 5, ....

The 167th trial division would have found divisor 991.

Not clear which method is better.

Dividing by small $p$ is faster than squaring mod $c$.

The $p-1$ method finds only 70 of the primes $\leq 1000$; trial division finds all 168 primes.
Given positive integer $n$, can compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z}/c$. Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

E.g. $c = 8597231219$: ...

$2^{27} \mod c = 134217728$;
$2^{54} \mod c = 134217728^2 \mod n = 935663516$;
$2^{55} \mod c = 1871327032$;
$2^{110} \mod c = 1871327032^2 \mod c = 1458876811$; ...;
$2^{232792560} - 1 \mod c = 5626089344$.

Easy extra computation (Euclid): \[\gcd\{5626089344, c\} = 991.\]

This $p-1$ method (1974 Pollard) quickly factored $c = 8597231219$. Main work: 27 squarings mod $c$.

Could instead have checked $c$'s divisibility by 2, 3, 5, ... The 167th trial division would have found divisor 991.

Not clear which method is better. Dividing by small $p$ is faster than squaring mod $c$.

The $p-1$ method finds only 70 of the primes $\leq 1000$; trial division finds all 168 primes.
Given positive integer $n$, compute $2^{232792560} - 1 \mod c$ using 41 operations in $\mathbb{Z}/c$.

Notation: $a \mod b = a - b \lfloor a/b \rfloor$.

given $c = 8597231219$: ... 

d $c = 134217728$; 

d $c = 134217728^2 \mod n$

$= 935663516$; 

d $c = 1871327032$; 

d $c = 1871327032^2 \mod c$

$= 1458876811$; 

$2^{110} - 1 \mod c = 5626089344$.

Extra computation (Euclid): 

$\gcd\{5626089344, c\} = 991$.

This $p - 1$ method (1974 Pollard) quickly factored $c = 8597231219$. Main work: 27 squarings mod $c$.

Could instead have checked $c$'s divisibility by 2, 3, 5, ... 

The 167th trial division would have found divisor 991.

Not clear which method is better.

Dividing by small $p$ is faster than squaring mod $c$. The $p - 1$ method finds only 70 of the primes $\leq 1000$; trial division finds all 168 primes.

Scale up to larger exponent $s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\}$: 

using 136 squarings mod $c$ find 2317 of the primes $\leq 10^7$.

Is a squaring mod $c$ faster than 125 trial divisions?

Or 

$s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}$: 

using 1438 squarings mod $c$ find 180121 of the primes $\leq 10^7$.

Is a squaring mod $c$ faster than 125 trial divisions?

Extra benefit: no need to store the primes.
Given positive integer \( n \),
\[ 2^{232792560} - 1 \mod c \]
computes in \( \mathbb{Z}/c \).
\[ a \mod b = a - b \lfloor a/b \rfloor. \]

219: \( \ldots \)
217728;
217728\(^2\) \mod n = 663516;
217728\(^2\) \mod c = 327032;
217728\(^2\) \mod c = 876811; \ldots ;
c = 5626089344.

Notation (Euclid):
\[ \gcd \{ \} = 991. \]

This \( p - 1 \) method (1974 Pollard) quickly factored \( c = 8597231219 \).
Main work: 27 squarings mod \( c \).

Could instead have checked \( c \)'s divisibility by 2, 3, 5, \ldots .
The 167th trial division would have found divisor 991.

Not clear which method is better.
Dividing by small \( p \) is faster than squaring mod \( c \).
The \( p - 1 \) method finds only 70 of the primes \( \leq 1000 \);
trial division finds all 168 primes.

Extra benefit:
opt need to store the primes.

Scale up to larger exponent \( s = \text{lcm}\{1, 2, 3, 4, \ldots , 100\} \):
using 136 squarings mod \( c \) find 2317 of the primes.

Is a squaring mod \( c \) faster than 17 trial divisions?
Or
\[ s = \text{lcm}\{1, 2, 3, 4, \ldots , 1000\} \]:
using 1438 squarings mod \( c \) find 180121 of the primes.

Is a squaring mod \( c \) faster than 125 trial divisions?
Extra benefit:
opt need to store the primes.
Given positive integer $n$, can compute $2^{2^{32792560}} - 1 \mod c$ using 41 operations in $\mathbb{Z} = c$.

Notation: $a \mod b = a - b \lfloor \frac{a}{b} \rfloor$.

\[ \begin{array}{l}
2^2 \mod c = 134217728; \\
2^5 \mod c = 134217728 \\
2^{11} \mod c = 935663516; \\
2^{27} \mod c = 1871327032; \\
2^{110} \mod c = 1871327032; \\
2^{232792560} - 1 \mod c = 5626089344. \\
\end{array} \]

Easy extra computation (Euclid):
$\gcd \{ 5626089344; c \} = 991.$

This $p - 1$ method (1974 Pollard) quickly factored $c = 8597231219$. Main work: 27 squarings mod $c$.

Could instead have checked $c$'s divisibility by 2, 3, 5, ... The 167th trial division would have found divisor 991.

Not clear which method is better. Dividing by small $p$ is faster than squaring mod $c$.

The $p - 1$ method finds only 70 of the primes $\leq 1000$; trial division finds all 168 primes.

Scale up to larger exponent $s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\}$:
using 136 squarings mod $c$ find 2317 of the primes $\leq 10^7$.

Is a squaring mod $c$ faster than 17 trial divisions?

Or

$s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}$:
using 1438 squarings mod $c$ find 180121 of the primes $\leq 10^{10}$.

Is a squaring mod $c$ faster than 125 trial divisions?

Extra benefit: no need to store the primes.
This \( p - 1 \) method (1974 Pollard) quickly factored \( c = 8597231219 \). Main work: 27 squarings mod \( c \).

Could instead have checked \( c \)'s divisibility by 2, 3, 5, \ldots.
The 167th trial division would have found divisor 991.

Not clear which method is better. Dividing by small \( p \) is faster than squaring mod \( c \).
The \( p - 1 \) method finds only 70 of the primes \( \leq 1000 \); trial division finds all 168 primes.

Scale up to larger exponent \( s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\} \):
using 136 squarings mod \( c \) find 2317 of the primes \( \leq 10^5 \).

Is a squaring mod \( c \) faster than 17 trial divisions?

Or
\( s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\} \):
using 1438 squarings mod \( c \) find 180121 of the primes \( \leq 10^7 \).

Is a squaring mod \( c \) faster than 125 trial divisions?

Extra benefit:
no need to store the primes.

Instead have checked

- divisibility by 2, 3, 5, ...
- trial division

Would have found divisor 991.

For which method is better.

- by small $p$
- than squaring mod $c$. $p-1$ method finds
  70 of the primes $\leq 1000$; trial division finds all 168 primes.

Scale up to larger exponent $s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\}$:

- using 136 squarings mod $c$ find 2317 of the primes $\leq 10^5$.
- Is a squaring mod $c$
  faster than 17 trial divisions?

Or

- $s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}$:
  using 1438 squarings mod $c$ find 180121 of the primes $\leq 10^7$.
- Is a squaring mod $c$
  faster than 125 trial divisions?

Extra benefit:

- no need to store the primes.

Plausible conjecture: if $K$ is

$$\exp\left(\frac{1}{12}q^{1/2}+o(1)\right)\log H \log \log H$$

then $p-1$ divides $\text{lcm}\{1, 2, 3, \ldots, K\}$

for $H=K^{1+o(1)}$ primes $p \leq H$.

Same if $p-1$ is replaced by

- order of 2 in $\mathbb{F}_p^\ast$.

So uniform random prime $p \leq H$

divides $2^{\text{lcm}\{1, 2, 3, \ldots, K\}}-1$ mod $c$

with probability $1 - K^{1+o(1)}$.

(1.4 ... - 1/2)

Similar time spent on trial division

finds far fewer primes for large $H$. 

Using $136$ squarings mod $c$ produce $2^{\text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\}}-1$ mod $c$.

Using $1438$ squarings mod $c$ find $2^{\text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}}-1$ mod $c$.

Similar time spent on trial division finds far fewer primes for large $H$. 

Extra benefit:

- no need to store the primes.
This \( p - 1 \) method (1974 Pollard) quickly factored \( c = 8597231219 \).

Main work: 27 squarings mod \( c \).

Could instead have checked \( c \)'s divisibility by 2, 3, 5, ….

The 167th trial division would have found divisor 991.

Not clear which method is better.

Dividing by small \( p \) is faster than squaring mod \( c \).

The \( p - 1 \) method finds only 70 of the primes \( \leq 1000 \);
trial division finds all 168 primes.

Scale up to larger exponent \( s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\} \):
using 136 squarings mod \( c \) find 2317 of the primes \( \leq 10^5 \).

Is a squaring mod \( c \) faster than 17 trial divisions?

Or

\( s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\} \):
using 1438 squarings mod \( c \) find 180121 of the primes \( \leq 10^7 \).

Is a squaring mod \( c \) faster than 125 trial divisions?

Extra benefit:
no need to store the primes.

Plausible conjecture: if \( K \) is
\( \exp \sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log H} \)
then \( p - 1 \) divides \( \text{lcm}\{1; 2; \ldots; K\} \) for \( H/K^{1+o(1)} \) primes.

Same if \( p - 1 \) is replaced by order of 2 in \( \mathbb{F}_p^* \).

So uniform random \( p \leq H \) divides \( 2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c \) with probability \( 1/K^{1+o(1)} \).

\((1.4 \ldots + o(1))K \) squarings mod \( c \) produce \( 2^{\text{lcm}\{1, 2, \ldots, 1000\}} \).

Similar time spent on trial division finds far fewer primes.
This method (1974 Pollard) quickly factored \( c = 8597231219 \). Main work: 27 squarings mod \( c \). Could instead have checked \( c \)'s divisibility by 2, 3, 5, \ldots.
The 167th trial division would have found divisor 991. Not clear which method is better. Dividing by small \( p \) is faster than squaring mod \( c \).
The \( p - 1 \) method finds only 70 of the primes \( \leq 1000 \); trial division finds all 168 primes.

Scale up to larger exponent \( s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\} \): using 136 squarings mod \( c \) find 2317 of the primes \( \leq 10^5 \).
Is a squaring mod \( c \) faster than 17 trial divisions?

Or
\[
s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}:
\]
using 1438 squarings mod \( c \) find 180121 of the primes \( \leq 10^7 \).
Is a squaring mod \( c \) faster than 125 trial divisions?

Extra benefit: no need to store the primes.

Plausible conjecture: if \( K \) is
\[
\exp \sqrt{\left( \frac{1}{2} + o(1) \right) \log H \log \log H}
\]
then \( p-1 \) divides \( \text{lcm}\{1, 2, \ldots, K\} - 1 \) for \( H/K^{1+o(1)} \) primes \( p \leq H \). Same if \( p-1 \) is replaced by order of 2 in \( \mathbb{F}_p^* \).

So uniform random prime \( p \leq H \) divides \( 2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \) with probability \( 1/K^{1+o(1)} \).
\[
(1.4 \ldots + o(1))K \text{ squarings produce } 2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c.
\]
Similar time spent on trial division finds far fewer primes for large \( H \).
Scale up to larger exponent 
$s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 100\}$:
using 136 squarings mod $c$
find 2317 of the primes $\leq 10^5$.

Is a squaring mod $c$
faster than 17 trial divisions?

Or 
$s = \text{lcm}\{1, 2, 3, 4, 5, \ldots, 1000\}$:
using 1438 squarings mod $c$
find 180121 of the primes $\leq 10^7$.

Is a squaring mod $c$
faster than 125 trial divisions?

Extra benefit:
no need to store the primes.

Plausible conjecture: if $K$ is
$\exp \sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log H}$
then $p-1$ divides $\text{lcm}\{1, 2, \ldots, K\}$
for $H/K^{1+o(1)}$ primes $p \leq H$.
Same if $p-1$ is replaced by
order of 2 in $\mathbf{F}_p^*$.

So uniform random prime $p \leq H$
divides $2^{\text{lcm}\{1,2,\ldots,K\}} - 1$
with probability $1/K^{1+o(1)}$.

$(1.4 \ldots + o(1))K$ squarings mod $c$
produce $2^{\text{lcm}\{1,2,\ldots,K\}} - 1 \mod c$.

Similar time spent on trial division
finds far fewer primes for large $H$. 

Plausible conjecture: if $K$ is
$$\exp \sqrt{\left( \frac{1}{2} + o(1) \right) \log H \log \log H}$$
then $p-1$ divides $\text{lcm}\{1, 2, \ldots, K\}$
for $H/K^{1+o(1)}$ primes $p \leq H$.

Same if $p-1$ is replaced by
order of 2 in $\mathbb{F}_p^*$.

So uniform random prime $p \leq H$
divides $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1$
with probability $1/K^{1+o(1)}$.

$(1.4 \ldots + o(1))K$ squarings mod $c$
produce $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c$.

Similar time spent on trial division
finds far fewer primes for large $H$.

Safe primes
This means numbers are easy
to factor if their factors $p_i$
have smooth $p_i - 1$.

To construct hard instances
avoid such factors – that's it?

ANSI does recommend
using “safe primes”, i.e.,
primes of the form $2p' + 1$
when generating RSA moduli.

This does not help against the
NFS nor against the following
algorithms.

Final:
This method does not make
lcm
{1, 2, 3, 4, 5, \ldots, 100}:
6 squarings mod $c$
717 of the primes $\leq 10^5$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 1000}:
38 squarings mod $c$
121 of the primes $\leq 10^7$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 10000}:
125 squarings mod $c$
find 231 of the primes $\leq 10^7$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 100000}:
250 squarings mod $c$
find 217 of the primes $\leq 10^9$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 1000000}:
500 squarings mod $c$
find 121 of the primes $\leq 10^{11}$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 10000000}:
1000 squarings mod $c$
find 51 of the primes $\leq 10^{13}$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 100000000}:
2000 squarings mod $c$
find 21 of the primes $\leq 10^{15}$.

$lcm$
{1, 2, 3, 4, 5, \ldots, 1000000000}:
4000 squarings mod $c$
find 9 of the primes $\leq 10^{17}$.
Plausible conjecture: if $K$ is
\[
\exp \sqrt{\frac{1}{2} + o(1)} \log H \log \log H
\] then $p-1$ divides $\text{lcm}\{1, 2, \ldots, K\}$ for $H/K^{1+o(1)}$ primes $p \leq H$.

Same if $p - 1$ is replaced by
order of 2 in $\mathbb{F}_p^*$.

So uniform random prime $p \leq H$
divides $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1$
with probability $1/K^{1+o(1)}$.

$(1.4 \ldots + o(1))K$ squarings mod $c$
produce $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c$.

Similar time spent on trial division
finds far fewer primes for large $H$.

Safe primes
This means numbers are easy
to factor if their factors $p_i$
have smooth $p_i - 1$.

To construct hard instances
avoid such factors – that's it?

ANSI does recommend
using “safe primes”,
primes of the form $2p' + 1$
when generating RSA moduli.

This does not help against the
NFS nor against the following
algorithms.

Safe primes
This means numbers are easy
to factor if their factors $p_i$
have smooth $p_i - 1$.

To construct hard instances
avoid such factors – that's it?

ANSI does recommend
using “safe primes”,
primes of the form $2p' + 1$
when generating RSA moduli.

This does not help against the
NFS nor against the following
algorithms.
Plausible conjecture: if $K$ is

\[ \exp \sqrt{\left( \frac{1}{2} + o(1) \right) \log H \log \log H} \]

then $p - 1$ divides $\text{lcm}\{1, 2, \ldots, K\}$ for $H/K^{1+o(1)}$ primes $p \leq H$.

Same if $p - 1$ is replaced by order of 2 in $\mathbb{F}_p^*$.

So uniform random prime $p \leq H$ divides $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1$

with probability $1/K^{1+o(1)}$.

$(1.4 \ldots + o(1))K$ squarings mod $c$ produce $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c$.

Similar time spent on trial division finds far fewer primes for large $H$.

Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it.

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it.

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.
Plausible conjecture: if $K$ is 
\[ \exp \sqrt{\left( \frac{1}{2} + o(1) \right) \log H \log \log H} \]
then $p-1$ divides $\text{lcm}\{1, 2, \ldots, K\}$ for $H/K^{1+o(1)}$ primes $p \leq H$.

Same if $p-1$ is replaced by order of 2 in $\mathbb{F}_p^*$. 

So uniform random prime $p \leq H$ divides $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1$
with probability $1/K^{1+o(1)}$.

$(1.4 \ldots + o(1))K$ squarings mod $c$ produce $2^{\text{lcm}\{1, 2, \ldots, K\}} - 1 \mod c$.

Similar time spent on trial division finds far fewer primes for large $H$.

Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$
when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.
Plausible conjecture: if \( K \) is
\( \exp\left(\frac{1}{2} + o(1)\right)\log H \log \log H \)
then \( p - 1 \) divides \( \text{lcm}\{1, 2, \ldots, K\} \)
for \( H = K^{1+o(1)} \) primes \( p \leq H \).

If \( p - 1 \) is replaced by \( 2 \) in \( \mathbb{F}_p^* \).

Uniform random prime \( p \leq H \)
\( 2^{\text{lcm}\{1,2,\ldots,K\}} - 1 \)
with probability \( 1/K^{1+o(1)} \).

\((1+o(1))K \) squarings mod \( c \)
\( 2^{\text{lcm}\{1,2,\ldots,K\}} - 1 \mod c \).

The time spent on trial division finds far fewer primes for large \( H \).

Safe primes

This means numbers are easy to factor if their factors \( p_i \)
have smooth \( p_i - 1 \).

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using “safe primes”, i.e.,
primes of the form \( 2p' + 1 \)
when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

The \( p + 1 \) factorization method
(1982 Williams)

Define \((X;Y) \in \mathbb{Q} \times \mathbb{Q}\) as the
\(232792560\)th multiple of \((3/5, 4/5)\) in the group \(\text{Clock}(\mathbb{Q})\).

The integer \( S = 2^{232792560}X \) is divisible by
82 of the primes \( \leq 10^3 \);
223 of the primes \( \leq 10^4 \);
455 of the primes \( \leq 10^5 \);
720 of the primes \( \leq 10^6 \);
etc.
Plausible conjecture: if $K$ is $\exp q^{1/2} + o(1)$
$log H \log \log H$
lcm{$\{1, 2, \ldots, K\}$} primes $p \leq H$.
then $p - 1$ divides lcm{$1; 2; \ldots; K$}
$p - 1$ is replaced by
the order of 2 in $F^*$ $p$
uniform random prime $p \leq H$
divides $2 \lcm{1; 2; \ldots; K} - 1$ with probability $1/K^{1+o(1)}$.

(1): $K$ squarings mod $c$
produce $2 \lcm{1; 2; \ldots; K} - 1$ mod $c$.

Similar time spent on trial division
finds far fewer primes for large $H$.

Safe primes
This means numbers are easy
to factor if their factors $p_i$
have smooth $p_i - 1$.
To construct hard instances
avoid such factors – that’s it?

ANSI does recommend
using “safe primes”, i.e.,
primes of the form $2p' + 1$
when generating RSA moduli.
This does not help against the
NFS nor against the following
algorithms.

The $p + 1$ factorization
(1982 Williams)
Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the
232792560th multiple of $(3/5, 4/5)$ in the group Clock$(\mathbb{Q})$.
The integer $S_2 = 5^{232792560}X$
is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.
Plausible conjecture: if $K$ is $\exp q \frac{1}{2} + o(1)$ then $p - 1$ divides $\text{lcm}\{1; 2; \ldots; K\}$ for $H = K^{1 + o(1)}$ primes $p \leq H$.

Same if $p - 1$ is replaced by the order of 2 in $F_p^*$. So uniform random prime $p \leq H$ divides $2^{\text{lcm}\{1; 2; \ldots; K\}} - 1$ with probability $1 - K^{1 + o(1)}$.

\[(1 + o(1) + \ldots + o(1))\] $K$ squarings mod $c$ produce $2^{\text{lcm}\{1; 2; \ldots; K\}} - 1$ mod $c$.

Safe primes
This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

The $p + 1$ factorization method
(1982 Williams)
Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the $2^{32792560}$th multiple of $(3 = 5; 4 = 5)$ in the group Clock($\mathbb{Q}$).

The integer $S_2 = 5^{232792560}X$ is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.
Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using “safe primes”, i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

The $p + 1$ factorization method

(1982 Williams)

Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the $232792560$th multiple of $(3/5, 4/5)$ in the group $\text{Clock}(\mathbb{Q})$.

The integer $S_2 = 5^{232792560}X$ is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.
Safe primes
This means numbers are easy
to factor if their factors $p_i$
have smooth $p_i - 1$.
To construct hard instances
avoid such factors – that’s it?
ANSI does recommend
using “safe primes”, i.e.,
primes of the form $2p' + 1$
when generating RSA moduli.
This does not help against the
NFS nor against the following
algorithms.

The $p + 1$ factorization method
(1982 Williams)
Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the
232792560th multiple of
$(3/5, 4/5)$ in the group $\text{Clock}(\mathbb{Q})$.
The integer $S_2 = 5^{232792560} X$
is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.

Given an integer $c$, compute $5^{232792560} X \mod c$
and compute gcd with $c$, hoping to factor $c$.
Many $p'$s not found by
F∗ and $p + 1$ are found by
Clock($F_p$).
If $-1$ is not a square mod $p$ and $p + 1$ divides $2^{232792560}$
then $5^{232792560} X = 0$.

Proof: $p' \equiv 3 \pmod{4}$, so
$(4/5 + 3i = 5)$
so $(p + 1)(3 = 5 ; 4 = 5) = (0 ; 1)$
in the group $\text{Clock}(F_p)$
so $232792560(3 = 5 ; 4 = 5) = (0 ; 1)$. 
Safe primes

This means numbers are easy to factor if their factors \( p_i \) have smooth \( p_i - 1 \).

To construct hard instances avoid such factors – that’s it?

ANSI does recommend using "safe primes", i.e., primes of the form \( 2^{p'} + 1 \) when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

The \( p + 1 \) factorization method
(1982 Williams)

Define \((X, Y) \in \mathbb{Q} \times \mathbb{Q}\) as the 232792560th multiple of \((3/5, 4/5)\) in the group \(\text{Clock}(\mathbb{Q})\).

The integer \(S_2 = 5^{232792560}X\) is divisible by
82 of the primes \( \leq 10^3 \);
223 of the primes \( \leq 10^4 \);
455 of the primes \( \leq 10^5 \);
720 of the primes \( \leq 10^6 \);
extc.

Given an integer \(c\), compute \(5^{232792560}X \mod c\) and compute \(\gcd\) with \(c\), hoping to factor \(c\).

Many \(p\)'s not found by \(F^*p\) are found by \(\text{Clock}(Fp)\).

If \(-1\) is not a square mod \(p\) and \(p + 1\) divides 232792560 then \(5^{232792560}X \mod p = 0\).

Proof: \(p \equiv 3 \pmod{4}\), so \((4/5 + 3i/5)^p = 4/5\), and \(p + 1\) divides \(232792560\), so \(232792560(3/5, 4/5) = (0, 1)\) in the group \(\text{Clock}(Fp)\).
Safe primes

This means numbers are easy to factor if their factors $p_i$ have smooth $p_i - 1$.

To construct hard instances avoid such factors – that's it?

ANSI does recommend using "safe primes", i.e., primes of the form $2p' + 1$ when generating RSA moduli.

This does not help against the NFS nor against the following algorithms.

---

The $p+1$ factorization method

Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the $232792560$th multiple of $(3/5, 4/5)$ in the group $\text{Clock}(Q)$.

The integer $S_2 = 523792560 X$ is divisible by $82$ of the primes $\leq 10^3$.
$223$ of the primes $\leq 10^4$.
$455$ of the primes $\leq 10^5$.
$720$ of the primes $\leq 10^6$.

Many $p$'s not found by $F^*$ are found by $\text{Clock}(F_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides $232792560$ then $523792560 X \mod p = 0$.

Given an integer $c$, compute $523792560 X \mod c$ and compute gcd with $c$, hoping to factor $c$.

---

Proof: $p \equiv 3 \pmod{4}$, so $(4/5 + 3i/5)^p = 4/5 - 3i/5$.

so $(p+1)(3/5, 4/5) = (0, 1)$ in the group $\text{Clock}(F_p)$.

and $p + 1$ divides $232792560$.

The integer $S_2 = 523792560 X$ is divisible by $82$ of the primes $\leq 10^3$.
$223$ of the primes $\leq 10^4$.
$455$ of the primes $\leq 10^5$.
$720$ of the primes $\leq 10^6$.

Many $p$'s not found by $F^*$ are found by $\text{Clock}(F_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides $232792560$ then $523792560 X \mod p = 0$.

Given an integer $c$, compute $523792560 X \mod c$ and compute gcd with $c$, hoping to factor $c$.

---

(1982 Williams)
The $p + 1$ factorization method
(1982 Williams)

Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the 232792560th multiple of $(3/5, 4/5)$ in the group $\text{Clock}(\mathbb{Q})$.

The integer $S_2 = 5^{232792560}X$ is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.

Given an integer $c$,
compute $5^{232792560}X \mod c$
and compute gcd with $c$,
hoping to factor $c$.

Many $p$'s not found by $\mathbf{F}_p^*$
are found by $\text{Clock}(\mathbf{F}_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides 232792560 then $5^{232792560}X \mod p = 0$.

Proof: $p \equiv 3 \pmod{4}$, so
$(4/5 + 3i/5)^p = 4/5 - 3i/5$ and
so $(p + 1)(3/5, 4/5) = (0, 1)$
in the group $\text{Clock}(\mathbf{F}_p)$
so $232792560(3/5, 4/5) = (0, 1)$.
The $p + 1$ factorization method
(1982 Williams)
Define $(X, Y) \in \mathbb{Q} \times \mathbb{Q}$ as the
232792560th multiple of $(3 = 5; 4 = 5)$ in the group Clock($\mathbb{Q}$).

The integer $S_2 = 5^{232792560}X$ is divisible by
82 of the primes $\leq 10^3$;
223 of the primes $\leq 10^4$;
455 of the primes $\leq 10^5$;
720 of the primes $\leq 10^6$;
etc.

Given an integer $c$, compute $5^{232792560}X \mod c$ and compute gcd with $c$, hoping to factor $c$.

Many $p$'s not found by $F^*_p$ are found by Clock($F_p$).

If $-1$ is not a square mod $p$ and $p + 1$ divides 232792560 then $5^{232792560}X \mod p = 0$.

Proof: $p \equiv 3 \pmod{4}$, so $(4/5 + 3i/5)^p = 4/5 - 3i/5$ and so $(p + 1)(3/5, 4/5) = (0, 1)$ in the group Clock($F_p$)
so 232792560(3/5, 4/5) = (0, 1).

The elliptic-curve method
Stage 1: Point $P$ on $E$ over $\mathbb{Z} = c$, compute $R = sP$ for $s = \text{lcm}\{2, 3; \cdots; B_1\}$.
Stage 2: Small primes $B_1 < q_1; \cdots; q_k \leq B_2$ compute $R_i = q_iR$.

If order of $P$ on $E = F_p$ (same curve, reduce mod $p$) divides $sq_i$, then $R_i = (0, 1)$ (using Edwards).

Compute $\gcd\{c; Q_y(R_i)\}$. 
The $p+1$ factorization method

Define $(X;Y) \in \mathbb{Q} \times \mathbb{Q}$ as the $2^{32792560}$th multiple of $(3=5;4=5)$ in the group $\text{Clock}(\mathbb{Q})$.

The integer $S_2 = 2^{32792560}X$ is divisible by
1. 82 of the primes $\leq 10^3$;
2. 223 of the primes $\leq 10^4$;
3. 455 of the primes $\leq 10^5$;
4. 720 of the primes $\leq 10^6$;
5. etc.

Given an integer $c$, compute $5^{2^{32792560}}X \mod c$ and compute \text{gcd} with $c$, hoping to factor $c$.

Many $p$'s not found by $\mathbf{F}_p^*$ are found by $\text{Clock}(\mathbb{F}_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides $2^{32792560}$ then $5^{2^{32792560}}X \mod p = 0$.

Proof: $p \equiv 3 \pmod{4}$, so $(4/5 + 3i/5)^p = 4/5 - 3i/5$ and so $(p+1)(3/5,4/5) = (0,1)$ in the group $\text{Clock}(\mathbb{F}_p)$ so $2^{32792560}(3/5,4/5) = (0,1)$.

The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z} = c$, compute $R = sP$ for $s = \text{lcm}\{2,3,\ldots,B_1\}$.

Stage 2: Small primes $B_1 < q_1,\ldots,q_k \leq B_2$ compute $R_i = q_iR$.

If order of $P$ on $E$ (same curve, reduce mod $p$) divides $sq_i$, then $R_i = (0,1)$ (using Edwards).

Compute $\text{gcd}\{c,\mathbb{Q}_y(R_i)\}$. 

Given an integer $c$, compute $5^{232792560}X \mod c$ and compute gcd with $c$, hoping to factor $c$.

Many $p$'s not found by $F_p^*$ are found by $\text{Clock}(F_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides 232792560 then $5^{232792560}X \mod p = 0$.

Proof: $p \equiv 3 \pmod{4}$, so $(4/5 + 3i/5)^p = 4/5 - 3i/5$ and so $(p + 1)(3/5, 4/5) = (0, 1)$ in the group $\text{Clock}(F_p)$ so $232792560(3/5, 4/5) = (0, 1)$.

The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z} = c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes $B_1 < q_1, \ldots, q_k \leq B_2$ compute $R_i = q_i R$.

If order of $P$ on $E/\mathbb{F}_{p_i}$ (same curve, reduce mod $p_i$) divides $s q_i$, then $R_i = (0, 1)$ (using Edwards).

Compute $\gcd\{c, \prod y(R_i)\}$. 
Given an integer $c$, compute $5^{232792560} \times \mod c$ and compute gcd with $c$, hoping to factor $c$.

Many $p$'s not found by $\mathbb{F}_p^*$ are found by $\text{Clock}(\mathbb{F}_p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides $232792560$ then $5^{232792560} \times \mod p = 0$.

Proof: $p \equiv 3 \pmod 4$, so $(4/5 + 3i/5)^p = 4/5 - 3i/5$ and so $(p + 1)(3/5, 4/5) = (0, 1)$ in the group $\text{Clock}(\mathbb{F}_p)$ so $232792560(3/5, 4/5) = (0, 1)$.

The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes $B_1 < q_1, \ldots, q_k \leq B_2$ compute $R_i = q_iR$.

If order of $P$ on $E/\mathbb{F}_{p_i}$ (same curve, reduce mod $p_i$) divides $sq_i$, then $R_i = (0, 1)$ (using Edwards).

Compute $\gcd\{c, \prod y(R_i)\}$. 
Given an integer \( c \), compute \( 5^{232792560} \mod c \) and compute gcd with \( c \), hoping to factor \( c \).

Many \( p \)’s not found by \( F^* \) are found by \( \text{Clock}(F_p) \).

If \( -1 \) is not a square mod \( p \) and \( p + 1 \) divides \( 232792560 \) then \( 5^{232792560} \mod p = 0 \).

Proof:
\[
p \equiv 3 \pmod{4}, \quad (3i/5)^p = 4/5 - 3i/5 \quad \text{and } \quad (1)(3/5, 4/5) = (0, 1)
\]
\( \text{Group Clock}(F_p) \)
\[
232792560(3/5, 4/5) = (0, 1).
\]

The elliptic-curve method

Stage 1: Point \( P \) on \( E \) over \( \mathbb{Z}/c \), compute \( R = sP \) for \( s = \text{lcm}\{2, 3, \ldots, B_1\} \).

Stage 2: Small primes \( B_1 < q_1, \ldots, q_k \leq B_2 \) compute \( R_i = q_i R \).

If order of \( P \) on \( E/F_{p_i} \) (same curve, reduce mod \( p_i \)) divides \( sq_i \), then
\[
R_i = (0, 1) \quad \text{(using Edwards)}.
\]
Compute \( \text{gcd}\{c, \prod y(R_i)\} \).

Good news (for the attacker):
All primes \( \leq H \) found after reasonable number of curves.

Order of elliptic-curve group \( \in [p + 1 - 2\sqrt{p}; p + 1 + 2\sqrt{p}] \).

If a curve fails, try another.

Plausible conjecture: if \( B_1 \) is
\[
\exp\sqrt{\frac{1}{2}\log H \log\log H}
\]
then, for each prime \( p \leq H \), a uniform random curve mod \( p \) has chance \( \geq 1/B_1 + o(1) \) to find \( p \).

Find \( p \) using, \( \leq B_1 + o(1) \) curves;
\( \leq B_2 + o(1) \) squarings.

Time subexponential in \( H \).
Given an integer $c$, compute $5^{232792560} \mod c$ and compute $\gcd$ with $c$, hoping to factor $c$.

Many $p$'s not found by $F^* p$ are found by $\text{Clock}(F p)$.

If $-1$ is not a square mod $p$ and $p + 1$ divides $232792560$ then $5^{232792560} \mod p = 0$.

**Proof:** $p \equiv 3 \mod 4$, so $(4 = 5 + 3 i = 5)$ $p = 4 = 5 - 3 i = 5$ and so $(p + 1)(3 = 5; 4 = 5) = (0; 1)$ in the group $\text{Clock}(F p)$ so $232792560(3 = 5; 4 = 5) = (0; 1)$.

The elliptic-curve method

**Stage 1:** Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

**Stage 2:** Small primes $B_1 < q_1, \ldots, q_k \leq B_2$ compute $R_i = q_i R$.

If order of $P$ on $E/\mathbb{F}_{p_i}$ (same curve, reduce mod $p_i$) divides $sq_i$, then $R_i = (0, 1)$ (using Edwards).

Compute $\gcd\{c, \prod y(R_i)\}$.

Good news (for the attacker):
All primes $\leq H$ found after reasonable number of curves.

Order of elliptic-curve group $\in [p + 1 - 2\sqrt{p}, p]$.
If a curve fails, try another.

Plausible conjecture: if $B_1$ is $\exp \sqrt{(\frac{1}{2} + o(1)) \log \log H}$, then, for each prime $p \leq H$, a uniform random curve mod $p$ has chance $\geq 1/B_1$ to find $p$.

Find $p$ using, $\leq B_1$ curves; $\leq B_2^{2+o(1)}$ squarings.

Time subexponential in $H$.
The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for
$s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes
$B_1 < q_1, \ldots, q_k \leq B_2$
compute $R_i = q_i R$.

If order of $P$ on $E/\mathbb{F}_{p_i}$
(same curve, reduce mod $p_i$)
divides $sq_i$, then
$R_i = (0, 1)$ (using Edwards).

Compute gcd$c, \prod y(R_i)\}$.

Good news (for the attacker):
All primes $\leq H$ found after a reasonable number of curves.

Order of elliptic-curve group
$\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$.
If a curve fails, try another.

Plausible conjecture: if $B_1$ is
$\exp \sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log \log H}$
then, for each prime $p \leq H$, a uniform random curve mod $p$
has chance $\geq 1/B_1^{1+o(1)}$ to
Find $p$ using, $\leq B_1^{1+o(1)}$ curves;
$\leq B_2^{2+o(1)}$ squarings.
Time subexponential in $H$. 

The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes $B_1 < q_1, \ldots, q_k \leq B_2$ compute $R_i = q_i R$.

If order of $P$ on $E/\mathbb{F}_{p_i}$ (same curve, reduce mod $p_i$) divides $sq_i$, then $R_i = (0, 1)$ (using Edwards).

Compute $\gcd\{c, \prod y(R_i)\}$.

Good news (for the attacker):

All primes $\leq H$ found after reasonable number of curves.

Order of elliptic-curve group $\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$.

If a curve fails, try another.

Plausible conjecture: if $B_1$ is $\exp \sqrt{(1/2 + o(1)) \log H \log \log H}$ then, for each prime $p \leq H$, a uniform random curve mod $p$ has chance $\geq 1/B_1^{1+o(1)}$ to find $p$.

Find $p$ using, $\leq B_1^{1+o(1)}$ curves; $\leq B_1^{2+o(1)}$ squarings.

Time subexponential in $H$. 
The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Small primes $q_1, \ldots, q_k \leq B_2$ compute $R_i = q_i R$.

Order of elliptic-curve group $\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$. If a curve fails, try another.

Plausible conjecture: if $B_1$ is 

$$\exp \sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log H}$$

then, for each prime $p \leq H$, a uniform random curve mod $p$ has chance $\geq 1/B_1^{1+o(1)}$ to find $p$.

Find $p$ using, $\leq B_1^{1+o(1)}$ curves; $\leq B_1^{2+o(1)}$ squarings.

Time subexponential in $H$.

Good news (for the attacker): All primes $\leq H$ found after reasonable number of curves.

Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.
The elliptic-curve method

Stage 1: Point $P$ on $E$ over $\mathbb{Z}/c$, compute $R = sP$ for $s = \text{lcm}\{2, 3, \ldots, B_1\}$.

Stage 2: Small primes $B_1 < q_1 \leq q_k \leq B_2$ compute $R_i = q_iR$.

If order of $P$ on $E = \mathbb{F}_p$ (same curve, reduce mod $p_i$) divides $q_i^2$, then $R_i = (0; 1)$ (using Edwards).

Compute $\gcd\{c; y(R_i)\}$.

Good news (for the attacker):
All primes $\leq H$ found after reasonable number of curves.

Order of elliptic-curve group
$\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$.
If a curve fails, try another.

Plausible conjecture: if $B_1$ is
$\exp \sqrt{(\frac{1}{2} + o(1)) \log H \log \log H}$
then, for each prime $p \leq H$,
a uniform random curve mod $p$
has chance $\geq 1/B_1^{1+o(1)}$ to find $p$.

Find $p$ using, $\leq B_1^{1+o(1)}$ curves;
$\leq B_1^{2+o(1)}$ squarings.
Time subexponential in $H$.

Bad RSA randomness
2004 Bauer–Laurie:
checked 18000 PGP RSA keys;
found 2 keys sharing a factor.

“Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.
Good news (for the attacker): All primes $\leq H$ found after reasonable number of curves.

Order of elliptic-curve group $\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$. If a curve fails, try another.

Plausible conjecture: if $B_1$ is

$$\exp\sqrt{\left(\frac{1}{2} + o(1)\right) \log H \log \log H}$$

then, for each prime $p \leq H$, a uniform random curve mod $p$ has chance $\geq 1/B_1^{1+o(1)}$ to find $p$.

Find $p$ using, $\leq B_1^{1+o(1)}$ curves; $\leq B_1^{2+o(1)}$ squarings. Time subexponential in $H$.

Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.
Good news (for the attacker): All primes $\leq H$ found after reasonable number of curves.

Order of elliptic-curve group $\in [p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$. If a curve fails, try another.

Plausible conjecture: if $B_1$ is $\exp \sqrt{(\frac{1}{2} + o(1)) \log H \log \log H}$ then, for each prime $p \leq H$, a uniform random curve mod $p$ has chance $\geq \frac{1}{B_1^{1+o(1)}}$ to find $p$.

Find $p$ using, $\leq B_1^{1+o(1)}$ curves; $\leq B_1^{2+o(1)}$ squarings. Time subexponential in $H$.

Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.
Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

“Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.

2012.02.17 Heninger–Durumeric–Wustrow–Halderman announcement (USENIX Security 2012): checked $> 10^7$ SSL/SSH RSA keys; factored 24816 SSL keys, 2422 SSH host keys. “Almost all of the vulnerable keys were generated to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”
Good news (for the attacker):
All primes \(\leq H\) found after reasonable number of curves.

Order of elliptic-curve group \(\in [p + 1 + 2\sqrt{p}; p + 1 + 2\sqrt{p}]\).

If a curve fails, try another.

A plausible conjecture: if \(B_1\) is \(\log H \log \log H\) then, for each prime \(p \leq H\), a uniform random curve mod \(p\) has chance \(\geq 1/(B_1 + o(1))\) to find \(p\).

Find \(p\) using, \(\leq B_1 + o(1)\) curves; \(\leq B_2 + o(1)\) squarings.

Time subexponential in \(H\).

Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.


“Ron was wrong, Whit is right” (Crypto 2012): checked \(7 \cdot 10^6\) SSL/PGP RSA keys; found \(6 \cdot 10^6\) distinct keys; factored 12720 of those, thanks to shared prime factors.

checked \(> 10^7\) SSL keys; factored 24816 SSL keys, 2422 SSH host keys.

“Almost all of the vulnerable keys were generated by or for secure embedded devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”
Good news (for the attacker):
All primes \( \leq H \) found after reasonable number of curves.

Order of elliptic-curve group \( \in [p+1-2\sqrt{p}; p+1+2\sqrt{p}] \).

If a curve fails, try another.

Plausible conjecture: if \( B_1 \) is \( \exp q^{1/2} + o(1) \) \( \log H \log \log H \) then, for each prime \( p \leq H \), a uniform random curve mod \( p \) has chance \( \geq 1/B_1+o(1) \) to find \( p \).

Find \( p \) using, \( \leq B_1+o(1) \) curves; \( \leq B_2+o(1) \) squarings.

Time subexponential in \( H \).

Bad RSA randomness

2004 Bauer–Laurie:
checked 18000 PGP RSA keys; found 2 keys sharing a factor.

"Ron was wrong, Whit is right" (Crypto 2012): checked \( 7 \cdot 10^6 \) SSL/PGP RSA keys; found \( 6 \cdot 10^6 \) distinct keys; factored 12720 of those, thanks to shared prime factors.

2012.02.17 Heninger–Durumeric–Wustrow–Halderman
announcement (USENIX Security 2012):
checked \( >10^7 \) SSL/SSH RSA keys; factored 24816 SSL keys, 2422 SSH host keys.

“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”
Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.


“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”
Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked \(7 \cdot 10^6\) SSL/PGP RSA keys; found \(6 \cdot 10^6\) distinct keys; factored 12720 of those, thanks to shared prime factors.


“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”

These computations find \(q_2\) in \(p_1 q_1, p_2 q_2, p_3 q_3, p_4 q_2, p_5 q_5, p_6 q_6, \) and thus also \(p_2\) and \(p_4\).

Obvious: GCD computation.

Faster: scaled remainder trees.

Nice follow-up project: Do this with Taiwan citizen cards.

Online data base of RSA keys. These were generated on certified smart cards; should have good randomness.

But: student broke 103 keys.
Bad RSA randomness

2004 Bauer–Laurie: checked 18000 PGP RSA keys; found 2 keys sharing a factor.

2012.02.14 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter “Ron was wrong, Whit is right” (Crypto 2012): checked $7 \cdot 10^6$ SSL/PGP RSA keys; found $6 \cdot 10^6$ distinct keys; factored 12720 of those, thanks to shared prime factors.


“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”

These computations find $q_2$ in $p_1q_1; p_2q_2; p_3q_3; p_4q_2; p_5q_5; p_6q_6; \ldots$ and thus also $p_2$ and $p_4$.

Obvious: GCD computation.

Faster: scaled remainder trees.

Nice follow-up project: Do this with Taiwan citizen cards.

Online data base of RSA keys.

These were generated on certified smart cards; should have good randomness.

But: student broke 103 keys.
checked $>10^7$ SSL/SSH RSA keys; factored 24816 SSL keys, 2422 SSH host keys.

“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”

These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3$, $p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$. Obvious: GCD computation. Faster: scaled remainder trees.

Nice follow-up project: Do this with Taiwan citizen online data base of RSA keys.

These were generated on certified smart cards; should have good randomness. But: student broke 103 keys.
checked > $10^7$ SSL/SSH RSA keys; factored 24,816 SSL keys, 2,422 SSH host keys.

“Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider.”

These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3, p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$.

Obvious: GCD computation.
Faster: scaled remainder trees.

Nice follow-up project:
Do this with Taiwan citizen cards.
Online data base of RSA keys.

These were generated on certified smart cards; should have good randomness.

But: student broke 103 keys.
checked > 10^7 SSL/SSH RSA keys; factored 24,816 SSL keys, 2,422 SSH host keys.

"Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider."

These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3, p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$.

Obvious: GCD computation.
Faster: scaled remainder trees.

Nice follow-up project:
Do this with Taiwan citizen cards.

Online data base of RSA keys.
These were generated on certified smart cards; should have good randomness.
But: student broke 103 keys.
checked > 10^7 SSL/SSH RSA keys; factored 24816 SSL keys, 2422 SSH host keys.
"Almost all of the vulnerable keys were generated by and are used to secure embedded hardware devices such as routers and firewalls, not to secure popular web sites such as your bank or email provider."

These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3, p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$. Obvious: GCD computation. Faster: scaled remainder trees.

Nice follow-up project: Do this with Taiwan citizen cards. Online data base of RSA keys. These were generated on certified smart cards; should have good randomness. But: student broke 103 keys.

Closer look at the 119 primes $p_{29}$, $p_{101}$, $p_{11}$, $p_{92}$, $p_{110}$, $p_{117}$, $p_{111}$, $p_3$, $p_{108}$, $p_{71}$, $p_5$, $p_{65}$, $p_{100}$, $p_{78}$, $p_{112}$, $p_{17}$, $p_{104}$, $p_{35}$, $p_{36}$, $p_{49}$, $p_{70}$, $p_{12}$, $p_{118}$, $p_{57}$, $p_{61}$, $p_{76}$, $p_{113}$, $p_{40}$, $p_{84}$, $p_{99}$, $p_{22}$, $p_{107}$, $p_{26}$, $p_{34}$, $p_{89}$, $p_{80}$, $p_{95}$, $p_{90}$, $p_8$, $p_{37}$, $p_{82}$, $p_{85}$, $p_{116}$, $p_{43}$, $p_{97}$, $p_{98}$, $p_{38}$, $p_{106}$, $p_{103}$, $p_{105}$, $p_{114}$, $p_{23}$, $p_{46}$, $p_{60}$, $p_{11}$, $p_{24}$, $p_{44}$, $p_{56}$, $p_{2}$, $p_{52}$, $p_{48}$, $p_{17}$, $p_{10}$, $p_{31}$, $p_{72}$, $p_{91}$, $p_{88}$, $p_{53}$, $p_{96}$, $p_{79}$, $p_{75}$, $p_{67}$, $p_{86}$, $p_{25}$, $p_{42}$, $p_{10}$.
These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3, p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$. Obvious: GCD computation. Faster: scaled remainder trees.

Nice follow-up project:
Do this with Taiwan citizen cards. Online data base of RSA keys.

These were generated on certified smart cards; should have good randomness. But: student broke 103 keys.
These computations find $q_2$ in $p_1 q_1, p_2 q_2, p_3 q_3,$ $p_4 q_2, p_5 q_5, p_6 q_6$; and thus also $p_2$ and $p_4$.

Obvious: GCD computation.

Faster: scaled remainder trees.

Nice follow-up project:
Do this with Taiwan citizen cards.
Online data base of RSA keys.

These were generated on certified smart cards; should have good randomness.
But: student broke 103 keys.
These computations find $q_2$ in $p_1 q_1$, $p_2 q_2$, $p_3 q_3$, $p_4 q_2$, $p_5 q_5$, $p_6 q_6$; and also $p_2$ and $p_4$.

- GCD computation.
- Faster: scaled remainder trees.

- Nice follow-up project: Do this with Taiwan citizen cards.
- Online data base of RSA keys.
- These were generated on certified smart cards; should have good randomness.

But: student broke 103 keys.
These computations find $q_2$ in $p_1q_1; p_2q_2; p_3q_3; p_4q_2; p_5q_5; p_6q_6$; and thus also $p_2$ and $p_4$.

Obvious: GCD computation.

Faster: scaled remainder trees.

Nice follow-up project:
Do this with Taiwan citizen cards.

Online data base of RSA keys.

These were generated on certified smart cards; should have good randomness.

But: student broke 103 keys.

Prime $p_{110}$ appears 46 times.
These computations find $q_2$ in $p_1 q_1; p_2 q_2; p_3 q_3; p_4 q_2; p_5 q_5; p_6 q_6;$ and thus also $p_2$ and $p_4$. Obvious: GCD computation. Faster: scaled remainder trees. Nice follow-up project: Do this with Taiwan citizen cards. Online data base of RSA keys. These were generated on certified smart cards; should have good randomness. But: student broke 103 keys.

Prime $p_{110}$ appears 46 times.
Closer look at the 119 primes

Prime p110 appears 46 times

c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000002f9
Closer look at the 119 primes

Prime \( p_{110} \) appears 46 times

\[
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000002f9
\]

which is the next prime after \( 2^{511} + 2^{510} \).
Prime p110 appears 46 times
\[c0000000000000000000000000000000\]
\[00000000000000000000000000000000\]
\[00000000000000000000000000000000\]
\[00000000000000000000000000000000\]
\[00000000000000000000000000000002f9\]
which is the next prime after \(2^{511} + 2^{510}\).

Next up
\[c9242492249292499249492449242492\]
\[24929249924949244924249224929249\]
\[92494924492424922492249292499249492\]
\[4924249224922499249492449242424e5\]
Several other factors exhibit such a pattern.
Prime p110 appears 46 times
\[c0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000002f9\]
which is the next prime after \(2^{511} + 2^{510}\).

Next up
\[c9242492249292499249492449242492492\]
\[2492924992494924492424924929249\]
\[92494924492424922492924992494924\]
\[492424922492924992494924492424\]

Several other factors exhibit such a pattern.
Prime p110 appears 46 times
\[\text{c0000000000000000000000000000000000000000000000000000000000000002f9}\]
which is the next prime after \(2^{511} + 2^{510}\).

Next up
\[\text{c9242492249292499249492444924242492}\]
\[249292499249492444924249224929249249\]
\[92494924449242424922492924924949249\]
\[4924249224929249924942444924242442424e5\]

Several other factors exhibit such a pattern.

Prime generation
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
Prime p110 appears 46 times

\[
\begin{align*}
c00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
00000000000000000000000000000002f9
\end{align*}
\]

which is the next prime after \(2^{511} + 2^{510}\).

Next up

\[
\begin{align*}
c924249224929249924949244924249249249 \\
24929249924949244924249224929249249 \\
92494924492424922492924992494924924 \\
49242492249292499249492449242424e5
\end{align*}
\]

Several other factors exhibit such a pattern.

Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
Prime p110 appears 46 times
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000002f9
which is the next prime after
$2^{511} + 2^{510}$.

Next up
c9242492249292499249492449242492
24929249924949244924249224929249
92494924492424922492924992494924
92424922492924992494924492424e5
Several other factors exhibit such
a pattern.

Prime generation
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
Prime p110 appears 46 times

c00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000000002f9

which is the next prime after $2^{511} + 2^{510}$.

Next up

c9242492249292499249492449242492
24929249924949244924249224929249
92494924492424922492924992494924
9242492249292499249492449242424e5

Several other factors exhibit such a pattern.

Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.
Prime p110 appears 46 times

which is the next prime after $2^{511} + 2^{510}$.

Next up

Several other factors exhibit such a pattern.

Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.
Prime p110 appears 46 times
000000000000000000000000000002f9
which is the next prime after
$2^{511} + 2^{510}$.

Next up
000000000000000000000000000002f9

Several other factors exhibit such a pattern.

Prime generation
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.
110 appears 46 times.

Prime generation
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Factoring by trial division
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.
Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.

Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.
Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.
Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.
Prime generation

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, \ldots
Prime generation
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Factoring by trial division
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,
:::

Computing GCDs factored 105 moduli, of which 18 were new.
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.

Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, …
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,\ldots

Computing GCDs factored 105 moduli, of which 18 were new.
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, ...

Computing GCDs factored 105 moduli, of which 18 were new.
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,\ldots

Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by “trial division”.
Factored 4 more keys using patterns of length 9.
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.
Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, ...

Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by “trial division”.
Factored 4 more keys using patterns of length 9.
More factors by studying other keys and using lattices.
“Factoring RSA keys from certified smart cards: Coppersmith in the wild”
(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)
http://smartfacts.cr.yp.to/
Factoring by trial division

Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.

For every 32-bit word, swap the lower and upper 16 bits.

Fix the most significant two bits to 11.

Find the next prime greater than or equal to this number.

Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, ...
Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by “trial division”.
Factored 4 more keys using patterns of length 9.
More factors by studying other keys and using lattices.
“Factoring RSA keys from certified smart cards: Coppersmith in the wild”
(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)
http://smartfacts.cr.yp.to/

Bad RSA randomness 2017 – ROCA
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]
Factoring by trial division
Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
For every 32-bit word, swap the lower and upper 16 bits.
Fix the most significant two bits to 11.
Find the next prime greater than or equal to this number.
Do this for any pattern:
0, 1, 001, 010, 011, 100, 101, 110
00001, 00010, 00011, 00100, 00101, :::

Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by “trial division”.
Factored 4 more keys using patterns of length 9.
More factors by studying other keys and using lattices.
“Factoring RSA keys from certified smart cards:
Coppersmith in the wild”
(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)
http://smartfacts.cr.yp.to/

Bad RSA randomness 2017
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]
Computing GCDs factored 105 moduli, of which 18 were new.

Breaking RSA-1024 by “trial division”.

Factored 4 more keys using patterns of length 9.

More factors by studying other keys and using lattices.

“Factoring RSA keys from certified smart cards: Coppersmith in the wild”
(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)

http://smartfacts.cr.yp.to/

Bad RSA randomness 2017 – ROCA
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas

All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]
Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by “trial division”.
Factored 4 more keys using patterns of length 9.
More factors by studying other keys and using lattices.
“Factoring RSA keys from certified smart cards: Coppersmith in the wild”
(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)
http://smartfacts.cr.yp.to/

Bad RSA randomness 2017 – ROCA
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
$n \mod 2 = 1$
$n \mod 11 \in \{1, 10\}$
$n \mod 37 \in \{1, 10, 37\}$
$n \mod 97 \in \{1, 35, 36, 61, 62, 96\}$
$n \mod 331 \in \{1, 330\}$
These give $1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72$ possibilities of $n \mod L$, where
$L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331$, instead of
$1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800$
Computing GCDs factored 105 moduli, of which 18 were new.

Breaking RSA-1024 by "trial division".

Factored 4 more keys using patterns of length 9.

More factors by studying other keys and using lattices.

"Factoring RSA keys from certified smart cards: Coppersmith in the wild"

(with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)

http://smartfacts.cr.yp.to/

Bad RSA randomness 2017 – ROCA

M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]

These give \(1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72\) possibilities of \(n \mod L\), where

\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331\]

instead of \(1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800\)

Worse,

\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
Bad RSA randomness 2017 – ROCA

M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]

These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where

\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \text{ instead of} \]
\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]

Worse,

\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
Computing GCDs factored 105 moduli, of which 18 were new.
Breaking RSA-1024 by "trial division".
Factored 4 more keys using patterns of length 9.
More factors by studying other keys and using lattices.
"Factoring RSA keys from certified smart cards: Coppersmith in the wild" (with D.J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, N. van Someren)

Bad RSA randomness 2017 – ROCA
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]
These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where
\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \]
instead of
\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]
Worse,
\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]

These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where

\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \text{ instead of } 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]

Worse,

\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
Bad RSA randomness 2017 – ROCA
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 10, 37\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 331 \in \{1, 330\} \]
These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \)
possibilities of \( n \mod L \), where
\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \text{ instead of } \]
\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]
Worse,
\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
\[ n \in \{65537^i \mod L | i \in \mathbb{Z}\} \]
and 65537 has order 6 mod \( L \).
All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]

\[ n \mod 11 \in \{1, 10\} \]

\[ n \mod 37 \in \{1, 10, 37\} \]

\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]

\[ n \mod 331 \in \{1, 330\} \]

These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where

\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \text{ instead of} \]

\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]

Worse,

\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]

\[ n \in \{65537^i \mod L | i \in \mathbb{Z}\} \]

and 65537 has order 6 \( \mod L \).

If \( n = p \cdot q = 65537^i \mod L \) then likely

\[ p, q \in \{65537^i \mod L | i \in \mathbb{Z}\}. \]
M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas
All RSA keys generated by some Infineon smart cards satisfy
$n \mod 2 = 1$
$n \mod 11 \in \{1, 10\}$
$n \mod 37 \in \{1, 10, 37\}$
$n \mod 97 \in \{1, 35, 36, 61, 62, 96\}$
$n \mod 331 \in \{1, 330\}$
These give $1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72$
possibilities of $n \mod L$, where
$L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331$, instead of
$1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800$

Worse,
$n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\}$
$n \in \{65537^i \mod L | i \in \mathbb{Z}\}$
and 65537 has order 6 mod $L$.
If $n = p \cdot q = 65537^i \mod L$ then likely
$p, q \in \{65537^i \mod L | i \in \mathbb{Z}\}$.
There are more congruences where this holds.
Actually $L = \prod_{\ell < 702, \ell \text{prime}} \ell$. 
All RSA keys generated by some Infineon smart cards satisfy
\[ n \mod 2 = 1 \]
\[ n \mod 11 \in \{1, 10\} \]
\[ n \mod 37 \in \{1, 35, 36, 61, 62, 96\} \]
\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96, 14367385\} \]
\[ n \mod 331 \in \{1, 330\} \]

These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where
\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \]
instead of
\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]

Worse, \( n \equiv 65537^i \mod L \) and \( 65537 \) has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \), then likely \( p, q \in \{65537^i \mod L \mid i \in \mathbb{Z}\} \).

There are more congruences where this holds.

Actually \( L = \prod_{l<702, \text{prime}} l \).
All RSA keys generated by some Infineon smart cards satisfy

\[ n \mod 2 = 1 \]

\[ n \mod 11 \in \{1, 10\} \]

\[ n \mod 37 \in \{1, 10, 37\} \]

\[ n \mod 97 \in \{1, 35, 36, 61, 62, 96\} \]

\[ n \mod 331 \in \{1, 330\} \]

These give \( 1 \cdot 2 \cdot 3 \cdot 6 \cdot 2 = 72 \) possibilities of \( n \mod L \), where

\[ L = 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331, \]

instead of

\[ 1 \cdot 10 \cdot 36 \cdot 96 \cdot 330 = 11404800 \]

Worse,

\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]

\[ n \in \{65537^i \mod L | i \in \mathbb{Z}\} \]

and 65537 has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \) then likely

\[ p, q \in \{65537^i \mod L | i \in \mathbb{Z}\}. \]

There are more congruences where this holds.

Actually \( L = \prod_{l<702, l \text{ prime}} l \).

How do these turn into primes?

\[ \log_2 L \approx 971 \text{ and } \log_2 p = 1024, \]

so \( p = p' + k \cdot L \), where \( p \equiv p' \mod L \) and \( k \) with \( \gcd\{k, L\} = 1 \) and \( \log_2 k \approx 53 \) is random so that \( p \) is prime.

Same for \( q \).
There are more congruences where this holds. Actually \( L = \prod_{\ell < 702.4 \text{prime}} \ell \).

\begin{align*}
\text{If } n &= p \cdot q = 65537 \mod L \text{ then likely } \\
p, q &\in \{65537 \mod L \mid i \in \mathbb{Z}\}. \\
\text{and } 65537 \text{ has order 6 mod } L.
\end{align*}

\begin{align*}
\text{So } p &= p' + k \cdot L, \\
\text{and } \gcd\{k, L\} &= 1 \text{ and } \log_2 k \approx 53 \text{ is random so that } p \text{ is prime.}
\end{align*}

\begin{align*}
\text{Same for } q.
\end{align*}
Worse, 
\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \]
\[ \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]

\[ n \in \{65537^i \mod L | i \in \mathbb{Z}\} \]
and 65537 has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \)
then likely
\[ p, q \in \{65537^i \mod L | i \in \mathbb{Z}\}. \]

There are more congruences where this holds.
Actually \( L = \prod_{\ell < 702, \ell \text{prime}} \ell \).

How do these turn into primes?
\[ \log_2 L \approx 971 \text{ and } \log_2 p = 1024, \]
so \( p = p' + k \cdot L \),
where \( p \equiv p' \mod L \), and \( k \) with
\[ \gcd\{k, L\} = 1 \text{ and } \log_2 k \approx 53 \]
is random so that \( p \) is prime.
Same for \( q \).
Worse,
\[ n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1, 65537, 4878941, 8942297, 14367385, 24016035\} \]
\[ n \in \{65537^i \mod L | i \in \mathbb{Z}\} \]
and 65537 has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \) then likely
\[ p, q \in \{65537^i \mod L | i \in \mathbb{Z}\}. \]

There are more congruences where this holds.

Actually \( L = \prod_{\ell < 702, \ell \text{prime}} \ell \).

How do these turn into primes?
\[ \log_2 L \approx 971 \text{ and } \log_2 p = 1024, \]
so \( p = p' + k \cdot L \),
where \( p \equiv p' \mod L \), and \( k \) with \( \gcd\{k, L\} = 1 \) and \( \log_2 k \approx 53 \)
is random so that \( p \) is prime.

Same for \( q \).

Lenstra’s “Divisors in Residue Classes” finds prime factors of
the form \( p = u + k \cdot L \)
efficiently if \( L \geq n^{1/3} \).

Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4} \).
\[ \log_2 L > 970 > 683 > 2048/3. \]
Worse, \( n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1; 65537; 4878941; 8942297; 14367385; 24016035\} \) and \( 65537 \) has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \) then likely \( p, q \in \bar{\{65537^i \mod L \mid i \in \mathbb{Z}\}} \).

There are more congruences where this holds.

Actually \( L = \prod_{\ell < 702, \ell \text{prime}} \ell \).

How do these turn into primes?
\[
\log_2 L \approx 971 \text{ and } \log_2 p = 1024, \\
\text{so } p = p' + k \cdot L, \\
\text{where } p \equiv p' \mod L \text{, and } k \text{ with } \gcd\{k, L\} = 1 \text{ and } \log_2 k \approx 53 \\
is random so that } p \text{ is prime.} \\
\text{Same for } q.
\]

Lenstra’s “Divisors in Residue Classes” finds prime factors of the form \( p = u + k \cdot L \)
efficiently if \( L \geq n^{1/3} \).

Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4} \).

\[
\log_2 L > 970 > 683 > 2048/3.
\]
Worse, \( n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1; 65537; 4878941; 8942297; 14367385; 24016035\} \) 
\( n \in \hat{\mathbb{Z}}_{65537} \) \( i \mod L \mid i \in \mathbb{Z} \) and 65537 has order 6 mod \( L \).

If \( n = p \cdot q = 65537^i \mod L \) then likely \( p, q \in \hat{\mathbb{Z}}_{65537}^i \mod L | i \in \mathbb{Z} \).

There are more congruences where this holds.

Actually \( L = \mathcal{Q}^{< 702}; \) prime.

\[ \log_2 L \approx 971 \text{ and } \log_2 p = 1024, \] so \( p = p^i + k \cdot L, \) where \( p \equiv p_i \mod L, \) and \( k \) is random so that \( p \) is prime.

Same for \( q. \)

Lenstra’s “Divisors in Residue Classes” finds prime factors of \( L \) efficiently if \( L \geq n^{1/3}. \) Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4}. \)

How do these turn into primes?

\[ \log_2 L > 970 > 683 > 2048/3. \]

Each run is cheap, but there are many options for \( p^i, \) e.g. \( 65537^i \mod 23 \in \{\pm 1, \pm 2, \pm 3, \pm 4, \ldots, \pm 9, \pm 10, \pm 11\}. \)

Run Lenstra for all \( p^i \in \hat{\mathbb{Z}}_{65537}^i \mod L | i \in \mathbb{Z} \).

Full attack
Worse, $n \mod 2 \cdot 11 \cdot 37 \cdot 97 \cdot 331 \in \{1; 65537; 4878941; 8942297; 14367385; 24016035\}$.

$n \in \tilde{65537} \mod L \mid i \in \mathbb{Z}$. There are more congruences where this holds.

Actually $L = Q' < 702$; prime.

How do these turn into primes?

$log_2 L \approx 971$ and $log_2 p = 1024$, so $p = p' + k \cdot L$, where $p \equiv p' \mod L$, and $k$ with $\gcd\{k, L\} = 1$ and $log_2 k \approx 53$ is random so that $p$ is prime. Same for $q$.

Lenstra's “Divisors in Residue Classes” finds prime factors of the form $p = u + k \cdot L$ efficiently if $L \geq n^{1/3}$.

Coppersmith, Howgrave-Graham, and Nagaraj work for $L \geq n^{1/4}$.

$log_2 L > 970 > 683 > 2048/3$.

Full attack

Run Lensta for all $p' \in \{65537^i \mod L | i \in \mathbb{Z}\}$.

Each run is cheap, but there are many options for $p'$, e.g. $65537^i \mod 23 \in \{\pm 1, \pm 2, \pm 3, \pm 4, \ldots, \pm 9, \pm 10, \pm 11\}$. 

"..."
How do these turn into primes?

\[ \log_2 L \approx 971 \quad \text{and} \quad \log_2 p = 1024, \]
so \( p = p' + k \cdot L \),
where \( p \equiv p' \mod L \), and \( k \) with
\[ \gcd\{k, L\} = 1 \quad \text{and} \quad \log_2 k \approx 53 \]
is random so that \( p \) is prime.

Same for \( q \).

Lenstra’s “Divisors in Residue Classes” finds prime factors of
the form \( p = u + k \cdot L \)
efficiently if \( L \geq n^{1/3} \).
Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4} \).

\[ \log_2 L > 970 > 683 > 2048/3. \]

**Full attack**

Run Lensta for all \( p' \in \{65537^i \mod L | i \in \mathbb{Z}\} \).

Each run is cheap, but
there are many options for \( p' \), e.g. \( 65537^i \mod 23 \in \{\pm 1, \pm 2, \pm 3, \pm 4, \ldots, \pm 9, \pm 10, \pm 11\} \).
How do these turn into primes?

\( \log_2 L \approx 971 \) and \( \log_2 p = 1024 \), so \( p = p' + k \cdot L \), where \( p \equiv p' \mod L \), and \( k \) with \( \gcd\{k, L\} = 1 \) and \( \log_2 k \approx 53 \) is random so that \( p \) is prime.

Same for \( q \).

Lenstra’s “Divisors in Residue Classes” finds prime factors of the form \( p = u + k \cdot L \) efficiently if \( L \geq n^{1/3} \).

Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4} \).

\( \log_2 L > 970 > 683 > 2048/3 \).

**Full attack**

Run Lensta for all \( p' \in \{65537^i \mod L \mid i \in \mathbb{Z}\} \).

Each run is cheap, but there are many options for \( p' \), e.g. \( 65537^i \mod 23 \in \{\pm1, \pm2, \pm3, \pm4, \ldots, \pm9, \pm10, \pm11\} \).

But \( L \) is much larger than needed. So use \( L' \mid L \) which minimizes number of choices \( \times \) runtime.
How do these turn into primes?

log₂ L ≈ 971 and log₂ p = 1024, so p = p′ + k · L, where p ≡ p′ mod L, and k with gcd{k; L} = 1 and log₂ k ≈ 53 are chosen so that p is prime.

Same for q.

Lenstra’s “Divisors in Residue Classes” finds prime factors of the form p = u + k · L efficiently if L ≥ n^{1/3}.

Coppersmith, Howgrave-Graham, and Nagaraj work for L ≥ n^{1/4}.

970 > 683 > 2048/3.

Full attack

Run Lensta for all p′ ∈ \{65537^i \mod L \mid i \in \mathbb{Z}\}.

Each run is cheap, but there are many options for p′, e.g. 65537^i \mod 23 ∈ \{±1, ±2, ±3, ±4, . . . , ±9, ±10, ±11\}.

But L is much larger than needed. So use L′\mid L which minimizes number of choices × runtime.

What went wrong here?

It would have been OK to choose p′ as

p′ \equiv 2^{r_1} \mod 3
p′ \equiv 3^{r_2} \mod 5
p′ \equiv ... and p′ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives 2 · 4 · 6 · 10 · 12 = 5760 options.
How do these turn into primes?

\[
\log_2 p = 1024, \\
\log_2 L \approx 971, \quad \log_2 k \approx 53
\]

so \( p = p' + k \cdot L \), where \( p \equiv p' \mod L \), and \( k \) with \( \gcd\{k;L\} = 1 \) and \( \log_2 k \approx 53 \) is random so that \( p \) is prime.

Same for \( q \).

Lenstra's "Divisors in Residue Classes" finds prime factors of the form \( p = u + k \cdot L \) efficiently if \( L \geq n^{1/2} \).

Coppersmith, Howgrave-Graham, and Nagaraj work for \( L \geq n^{1/4} \).

\[
\log_2 L > 970 > 683 > 2048/3.
\]

Full attack

Run Lensta for all \( p' \in \{65537^i \mod L \mid i \in \mathbb{Z}\} \).

Each run is cheap, but there are many options for \( p' \), e.g. \( 65537^i \mod 23 \in \{\pm1, \pm2, \pm3, \pm4, \ldots, \pm9, \pm10, \pm11\} \).

But \( L \) is much larger than needed.

So use \( L' | L \) which minimizes number of choices \( \times \) runtime.

What went wrong?

It would have been OK to choose \( p' \) as

\[
p' \equiv 2^{r_1} \mod 3 \\
p' \equiv 3^{r_2} \mod 5 \\
p' \equiv 3^{r_3} \mod 7 \\
p' \equiv 2^{r_4} \mod 11 \\
p' \equiv 2^{r_5} \mod 13
\]

with \( r_i \) random and \( p' \) reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

\[
2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760 \text{ options.}
\]
Full attack

Run Lenstra for all \( p' \in \{65537^i \mod L | i \in \mathbb{Z}\} \).

Each run is cheap, but there are many options for \( p' \), e.g. \( 65537^i \mod 23 \in \{\pm 1, \pm 2, \pm 3, \pm 4, \ldots, \pm 9, \pm 10, \pm 11\} \).

But \( L \) is much larger than needed. So use \( L' | L \) which minimizes number of choices \( \times \) runtime.

What went wrong here?

It would have been OK to choose \( p' \) as

\[
\begin{align*}
p' &\equiv 2^{r_1} \mod 3 \\
p' &\equiv 3^{r_2} \mod 5 \\
p' &\equiv 3^{r_3} \mod 7 \\
p' &\equiv 2^{r_4} \mod 11 \\
p' &\equiv 2^{r_5} \mod 13
\end{align*}
\]

with \( r_i \) random and \( p' \) reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

\[
2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760 \text{ options.}
\]
Full attack

Run Lensta for all \( p' \in \{65537^i \mod L | i \in \mathbb{Z}\} \).

Each run is cheap, but there are many options for \( p' \), e.g. \( 65537^i \mod 23 \in \{\pm1, \pm2, \pm3, \pm4, \ldots, \pm9, \pm10, \pm11\} \).

But \( L \) is much larger than needed. So use \( L' | L \) which minimizes number of choices \( \times \) runtime.

What went wrong here?

It would have been OK to choose \( p' \) as
\[
\begin{align*}
p' &\equiv 2^{r_1} \mod 3 \\
p' &\equiv 3^{r_2} \mod 5 \\
p' &\equiv 3^{r_3} \mod 7 \\
p' &\equiv 2^{r_4} \mod 11 \\
p' &\equiv 2^{r_5} \mod 13
\end{align*}
\]
with \( r_i \) random and \( p' \) reconstructed using CRT.

Note: 2 and 3 are generators, so this gives
\[
2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760 \text{ options.}
\]
What went wrong here?

It would have been OK to choose $p'$ as

\[ p' \equiv 2^{r_1} \pmod{3}\]
\[ p' \equiv 3^{r_2} \pmod{5}\]
\[ p' \equiv 3^{r_3} \pmod{7}\]
\[ p' \equiv 2^{r_4} \pmod{11}\]
\[ p' \equiv 2^{r_5} \pmod{13}\]

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

\[ 2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760 \text{ options.} \]
What went wrong here?

It would have been OK to choose $p'$ as

$p' \equiv 2^{r_1} \mod 3$
$p' \equiv 3^{r_2} \mod 5$
$p' \equiv 3^{r_3} \mod 7$
$p' \equiv 2^{r_4} \mod 11$
$p' \equiv 2^{r_5} \mod 13$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

$2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760$ options.

It would have OK’ish but worse to choose $p'$ as

$p' \equiv 2^{r_1} \mod 3$
$p' \equiv 2^{r_2} \mod 5$
$p' \equiv 2^{r_3} \mod 7$
$p' \equiv 2^{r_4} \mod 11$
$p' \equiv 2^{r_5} \mod 13$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only

$2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880$ options.
What went wrong here?

It would have been OK to choose $p'$ as

$p' \equiv 2^{r_1} \mod 3$
$p' \equiv 3^{r_2} \mod 5$
$p' \equiv 3^{r_3} \mod 7$
$p' \equiv 2^{r_4} \mod 11$
$p' \equiv 2^{r_5} \mod 13$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

$2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760$ options.

It would have OK’ish but worse to choose $p'$ as

$p' \equiv 2^{r_1} \mod 3$
$p' \equiv 2^{r_2} \mod 5$
$p' \equiv 2^{r_3} \mod 7$
$p' \equiv 2^{r_4} \mod 11$
$p' \equiv 2^{r_5} \mod 13$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only

$2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880$ options.
What went wrong here?

It would have been OK to choose $p'$ as

\[ p' \equiv 2^{r_1} \mod 3 \]
\[ p' \equiv 3^{r_2} \mod 5 \]
\[ p' \equiv 3^{r_3} \mod 7 \]
\[ p' \equiv 2^{r_4} \mod 11 \]
\[ p' \equiv 2^{r_5} \mod 13 \]

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives

\[ 2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760 \text{ options.} \]

It would have OK'ish but worse to choose $p'$ as

\[ p' \equiv 2^{r_1} \mod 3 \]
\[ p' \equiv 2^{r_2} \mod 5 \]
\[ p' \equiv 2^{r_3} \mod 7 \]
\[ p' \equiv 2^{r_4} \mod 11 \]
\[ p' \equiv 2^{r_5} \mod 13 \]

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only

\[ 2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880 \text{ options.} \]
What went wrong here?

It would have been OK to choose $p'$ as

$$p' \equiv 2^{r_1} \mod 3$$
$$p' \equiv 3^{r_2} \mod 5$$
$$p' \equiv 3^{r_3} \mod 7$$
$$p' \equiv 2^{r_4} \mod 11$$
$$p' \equiv 2^{r_5} \mod 13$$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 and 3 are generators, so this gives $2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 5760$ options.

It is really bad to replace this by a single exponentiation and choose $p'$ as

$$p' \equiv 5477^{r} \mod 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$$

with $r$ random.

Note: The orders of 5477 modulo 3, 5, 7, 11, and 13 are 2, 4, 6, 2, and 6, but the powers are linked. Instead of $2 \cdot 4 \cdot 6 \cdot 2 \cdot 6 = 576$ options, this gives $lcm\{2, 4, 6, 2, 6\} = 12$ options.

It would have OK’ish but worse to choose $p'$ as

$$p' \equiv 2^{2r_1} \mod 3$$
$$p' \equiv 2^{r_2} \mod 5$$
$$p' \equiv 2^{r_3} \mod 7$$
$$p' \equiv 2^{r_4} \mod 11$$
$$p' \equiv 2^{r_5} \mod 13$$

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only $2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880$ options.
What went wrong here?

It would have OK’ish but worse to choose $p’$ as

$$p’ \equiv 2^r_1 \mod 3$$
$$p’ \equiv 2^r_2 \mod 5$$
$$p’ \equiv 2^r_3 \mod 7$$
$$p’ \equiv 2^r_4 \mod 11$$
$$p’ \equiv 2^r_5 \mod 13$$

with $r_i$ random and $p’$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only

$$2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880 \text{ options.}$$

It is really bad to replace this by a single exponentiation and choose

$$p’ \equiv 5477^r \mod 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$$

with $r$ random.

Note: The orders of 5477 modulo 3,5,7,11, and 13 are 2,4,6,2, and 6,6,6,6,6,6, but they are linked.

Instead of $2 \cdot 4 \cdot 6 \cdot 2 \cdot 6 = 576$, this gives $\text{lcm}\{2,4,6\} = 12$ options.
It would have OK’ish but worse to choose $p'$ as

\[
p' \equiv 2^{r_1} \mod 3
\]

\[
p' \equiv 2^{r_2} \mod 5
\]

\[
p' \equiv 2^{r_3} \mod 7
\]

\[
p' \equiv 2^{r_4} \mod 11
\]

\[
p' \equiv 2^{r_5} \mod 13
\]

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only

\[
2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880 \text{ options.}
\]

It is really bad to replace this by a single exponentiation and choose $p'$ as

\[
p' \equiv 5477^r \mod 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13
\]

with $r$ random.

Note:
The orders of 5477 modulo 3, 5, 7, 11, and 13 are 2, 4, 6, 2, and 6, but the powers are linked.

Instead of $2 \cdot 4 \cdot 6 \cdot 2 \cdot 6 = 576$, this gives \( \text{lcm}\{2, 4, 6, 2, 6\} = 12 \) options.
It would have OK’ish but worse to choose $p'$ as

\[ p' \equiv 2^{r_1} \mod 3 \]
\[ p' \equiv 2^{r_2} \mod 5 \]
\[ p' \equiv 2^{r_3} \mod 7 \]
\[ p' \equiv 2^{r_4} \mod 11 \]
\[ p' \equiv 2^{r_5} \mod 13 \]

with $r_i$ random and $p'$ reconstructed using CRT.

Note: 2 is not always a generator, this gives only $2 \cdot 4 \cdot 3 \cdot 10 \cdot 12 = 2880$ options.

It is really bad to replace this by a single exponentiation and choose $p'$ as

\[ p' \equiv 5477^r \mod 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \]

with $r$ random.

Note: The orders of 5477 modulo 3, 5, 7, 11, and 13 are 2, 4, 6, 2, and 6, but the powers are linked.

Instead of $2 \cdot 4 \cdot 6 \cdot 2 \cdot 6 = 576$
this gives $\text{lcm}\{2, 4, 6, 2, 6\} = 12$ options.