Factorization – a two year journey

Preview Release v0.104

Author: Essbee Vanhoutte
Date: June 2025

Preface

This paper is dedicated to my family and friends, among whom are my former managers who gave me the time and support to grow my interest in cryptography and eventually the math behind it. When I started this journey two years ago, as a high-school drop-out with limited math education, this seemed like an impossible mountain to scale. This paper documents my findings, and proposes an improvement to the quadratic sieve algorithm.

Chapters

I. Integer to Quadratic

In attempting to break factorization, it is important to find the correct representation of the problem.

To describe a problem, one of your first steps will be to define some type of algebraic expression with variables representing the solutions you are trying to find.

The Number Field Sieve algorithm[1] is a great example of this, where the very beginning of the algorithm is the polynomial selection step.

I would like to preface this chapter, by giving a short review of the steps that led me to my representation of quadratics of the form:

$x^2 + y_0x + N = 0$ or $x^2 + y_1x - N = 0$. (We ignore the quadratic term's coefficient until later)

This entire paper is about semi−primes. A semi−prime is a composite number, which has exactly 2 unique prime factors (excluding itself and 1). For the security of the RSA algorithm, this is the only format that matters. Numbers with more factors then this will actually weaken the RSA algorithm[2] (since it reduces the possible search space for factors).
You will need to make your own generalizations if you want to apply this for composites with more factors.

One example of a semi−prime I've used often in the last two years is:

$p = 41$ and $q = 107 \Rightarrow N = 41 \times 107 = 4387$ (Where semi-prime N is the product of factors p and q)

One reason I would often use the same semi−prime while doing research, is that you eventually become so familiar with its structure, that any patterns immediately stand out. However, I always make sure my findings generalize to any other semi−prime as well. In this paper I will exclusively use $p = 41$ and $q = 107$. You can use my proof of concepts to verify these findings against arbitrary semi−primes. I am not an educated Mathematician, hence I will not even attempt to write a paper with algebraic letter−soup.

Around two years ago, one of the first things I very quickly started to zero in on, was using modular reduction on N. My initial thought process was that perhaps I could find some type of pattern in the remainders that would let me construct an algorithm.

For example if $N = 4387$:

4387	=	47	mod 70
4387	=	40	mod 69
4387	=	35	mod 68
4387	=	32	mod 67
4387	=	31	mod 66 (+/− $\sqrt{N}$)
4387	=	32	mod 65
4387	=	35	mod 64
4387	=	40	mod 63
4387	=	47	mod 62

The rounded $\sqrt{N}$ (square root of N) is a point of symmetry for the remainders.
In the above example, starting from mod 66, we get the following sequence as remainders in both directions if we increase or decrease the modulus:

$31, 32, 35, 40, 47, .., \ = 31 + \sum_{i=1}^{N} 2i - 1$

Any trained mathematician will immediately recognize this sequence as simply 31 + the squares.
This is as expected, since we are modulo reducing a number close to its square root.

Now we can rephrase factorization as finding some modulus m, such that $N = 0 \bmod m$ (aka m is a divisor of N).
And we also know that the remainders can be calculate by simply adding squares to the initial remainder at the square root of N.

Around two years ago, I was maybe a couple of weeks in to my research, when I made these realizations and thought :

“Okay, I can do this, this can't be very hard if I know by how much the remainder increases or decreases with each increment or decrement of the modulus, hence it should be easy to figure out when it hits 0 mod m!”.

Math is full of problems that are very deceptive. Intuitively it would seem like a straight forward mathematical tool to solve this should exist, but the reality of it is much more complicated. At the surface it looks like a shallow pond, but below the surface, are depths of complexity deeper then the deepest oceans.

Thus if we take $N \bmod m$, we need to figure out a way to adjust modulus m such that the remainder reaches 0 mod m. At first I would mess around with some algebraic expressions which had an unknown variable in the modulus. Since to make the remainder change in value, we have to increment or decrement the modulus. However, soon after I was able to generalize it to this, removing any unknowns in the modulus:

$x^2 + yx + N = 0 \bmod m$

The root, x, would represent the smallest factor. In this case 41. And we negate the x so it becomes negative: −41.
The coefficient y, would represent the sum of both factors, in this case 41 + 107 or 148.

Thus we get: $(-41)^2 + 148 \times -41 + 4387 = 0$

or simplified:

$41^2 - 148 \times 41 + 4387 = 0$

And of−course this will also equal 0 in any mod m.

Note: My motivation for making the root, x, negative sign, is that when we modulo reduce it, it represents the distance to the factor from mod m. This is related to how I came up originally to represent factorization this way. I went a little bit into the background on this in my original paper, but truth is, its not very relevant, it's simply one of many ways to represent this problem.

We can also flip the signs in the equation, but then the coefficient instead of being $p + q$ becomes $p - q$.

If $p + q = 148$ then $p - q = -66$

And we get: $41^2 + 66 \times 41 - 4387 = 0$

Or combined: $41^2 - 148 \times 41 + 4387 = 41^2 + 66 \times 41 - 4387$

We have two unknowns, the roots and the linear coefficients (ignoring the quadratic term's coefficient for now). However, isolating the linear coefficient on both sides - which we can do if they share the same root - of the equation and squaring it mod N we get:

$148^2 = 66^2 \bmod 4387$ (we can ignore the signs when squaring)

And because these two squares are congruent mod N, we can take the gcd (greatest common divisor) of their difference:

$\gcd(148 + 66, 4387) = 107$
$\gcd(148 - 66, 4387) = 41$

Note: Calculating the greatest common divisor can be done very quickly using the Euclidean algorithm[3].

Because of this property, just finding the linear coefficients of a quadratic will result in the factorization of N.
Hence:

$x^2 + y_0x + N = x^2 + y_1x - N$

We can factor N by finding y_0 and y_1 and taking the gcd of their difference.
This is very similar to what Fermat's factorization method[4] does.

II. Fermat's factorization method

I feel it is important to now explain Fermat's factorization method, so we can draw some parallels with my own findings, which accidentally ended up converging with Fermat's factorization method. But this just shows how fundamental this is to the factorization problem.

In its most basic form the procedure is as this:

1. Calculate the rounded $\sqrt{N}$.
2. Starting from $x = \sqrt{N}$, calculate $y = x^2 - N$ (do so in a loop while incrementing x).

3. If y is also a square, calculate the greatest common divisor (gcd) on the difference.

Taking 4387 as an example:

Step 1 (Calculate the +/− square root of N):

$\sqrt{4387} = 66$ (rounded)

Step 2 (Starting from the square root calculate $y = x^2 - N$):

$66^2 - 4387 = -31$ (not a square)
$67^2 - 4387 = 102$ (not a square)
$68^2 - 4387 = 237$ (not a square)
$69^2 - 4387 = 374$ (not a square)
$70^2 - 4387 = 513$ (not a square)
$71^2 - 4387 = 654$ (not a square)
$72^2 - 4387 = 797$ (not a square)
$73^2 - 4387 = 942$ (not a square)
$74^2 - 4387 = \sqrt{1089} = 33$ (square)

Step 3 (If y is also a square, calculate the gcd on the difference.):

$74^2 = 33^2 \bmod 4387$

$\gcd(74 + 33, 4387) = 107$
$\gcd(74 - 33, 4387) = 41$

You may notice that 74 and 33 are simply 148 and 66 divided by 2.
This brings us to another important point. Many such square relations can be found mod N. What will be different is the amount of times N is in−between both squares.

$148^2 = 66^2 + 4 \times 4387$ (four times N in between)
$74^2 = 33^2 + 4387$ (one times N in between)

This difference of N, is related to the coefficient ot the quadratic term. More on this in chapter VI.

Do be aware, there are two types of square relations mod N. One which will yield a trivial factorization (1 or N) and one which will yield a non−trivial factorization (a prime factor of N). The square relations that will always yield a trivial factorization are of the form:

$a^2 = (N - a)^2 \bmod N$

We are not interested in theses square relations.

Both Quadratic Sieve[5] and Number Field Sieve - the current fastest factorization algorithms - are more elaborate ways of finding these square relations mod N. Both algorithms were invented by Carl Pomerance[6]. These algorithms are now almost 40 years old, and not much progress aside from a handful of tweaks to these algorithms has been made since. In my opinion, this is not acceptable. A problem as important as factorization should not go without major progress for 40 years. And simply hoping Quantum computing will solve everything is foolish. Thinking as such is the same as thinking AI will replace everything. It is but an excuse to stop trying. We should never stop trying. The day we stop trying, we surrender ourselves to ignorance.

III. Quadratic to Quadratic congruence

Going back to representing factorization as the following Quadratic:

$x^2 + yx + N = 0$

Finding a root and (linear) coefficient solution in the integers to this is very hard.
A lot of my research efforts have gone into this.
One approach is to create "fragments" of a possible solution by reducing everything to mod $p_0,p_1,p_2,..,p_{n-1}$ (where p is prime) and finding integer solutions mod $p_0,p_1,p_2,..,p_{n-1}$ and then combining them. This in essence turns the problem into a subset sum[7] type of problem, because then it becomes a matter of which "fragments", aka solutions mod p_i , to combine.

Lets dig in.

Our representation now becomes the same quadratic but with modular reduction:

$x^2 + yx + N = 0 \bmod m$

Instead of finding solutions in the integers, we reduce the scope to mod m.

Example:

$N = 4387 = 6 \bmod 13$
$p = 41 = 2 \bmod 13$
$q = 107 = 3 \bmod 13$

The residue of N mod 13 is the residue of pq mod 13 ($2 \times 3 = 6 \bmod 13$).

The residue of y mod 13 is $p + q$ mod 13 ($2 + 3 = 5$ mod 13).
If the coefficient, y, is 5 mod 13 and we know our residue of N mod 13 is 6 then only $2 + 3$ and $3 + 2$ can be our two residues for p and q mod 13.

Thus the root, x, is either -3 or -2. (remember we negate the root)

Plugging in for $y = 5$ and $x = -3$ or -2 in mod 13:

$-2^2 + 5 \times -2 + 4387 = 0$ mod 13
$\Rightarrow 4 - 10 + 6 = 0$ mod 13

$-3^2 + 5 \times -3 + 4387 = 0$ mod 13
$\Rightarrow 9 - 15 + 6 = 0$ mod 13

In real life we don't know y is 5 (since we do not know $p + q$).
We do know N mod 13 is 6.
Thus all possible coefficient solutions y mod 13 can be enumerated by summing up each of the two residues mod 13 that multiply to 6.

A coefficient solution y can also be told to exist, if for a given coefficient value y a root solution x exists. This can be trivially determined using the Legendre symbol[8] without actually having to find roots.

All root x and coefficient y solutions mod 13 that solve the quadratic congruence:

y: 1 x: $-5 \Rightarrow -5^2 + 1 \times -5 + 4387 = 0$ mod 13
y: 1 x: $-9 \Rightarrow -9^2 + 1 \times -9 + 4387 = 0$ mod 13
y: 5 x: $-2 \Rightarrow -2^2 + 5 \times -2 + 4387 = 0$ mod 13
y: 5 x: $-3 \Rightarrow -3^2 + 5 \times -3 + 4387 = 0$ mod 13
y: 6 x: $-7 \Rightarrow -7^2 + 6 \times -7 + 4387 = 0$ mod 13
y: 6 x: $-12 \Rightarrow -12^2 + 6 \times -12 + 4387 = 0$ mod 13
y: 7 x: $-1 \Rightarrow -1^2 + 7 \times -1 + 4387 = 0$ mod 13
y: 7 x: $-6 \Rightarrow -6^2 + 7 \times -6 + 4387 = 0$ mod 13
y: 8 x: $-10 \Rightarrow -10^2 + 8 \times -10 + 4387 = 0$ mod 13
y: 8 x: $-11 \Rightarrow -11^2 + 8 \times -11 + 4387 = 0$ mod 13
y: 12 x: $-4 \Rightarrow -4^2 + 12 \times -4 + 4387 = 0$ mod 13
y: 12 x: $-8 \Rightarrow -8^2 + 12 \times -8 + 4387 = 0$ mod 13

IV. Combining modular linear coefficient solutions

Note: This is simply Chinese Remainder Theorem. I found this independently while messing around with inverses. Additionally all these modular inverse calculations can be compressed into just a single modular inverse, which I should have realized sooner. I probably could leave this chapter out of the paper, but I will leave it in here for historical purposes. In a way it is good that I am stumbling on existing tricks independently, albeit frustrating as it is.

Now that we can find (linear) coefficient solutions mod p_i. Let us calculate coefficient solutions mod $p_0, p_1, p_2, .., p_{n-1}$ now and combine the results into mod m (where $m = p_0 \times p_1 \times p_2 \times \ldots \times p_{n-1}$)

y mod 3 = { 1, 2 }
y mod 5 = { 2, 3 }
y mod 7 = { 0, 1, 6 }
y mod 11 = { 1, 2, 5, 6, 9, 10 }
y mod 13 = { 1, 5, 6, 7, 8, 12 }

These are the solutions mod 3,5,7,11,13 that solve the coefficient of the linear term in: $x^2 + yx + N = 0$ mod m

If we calculate the solution set mod 15015 we get:

y mod 15015 = { 83, 97, 98, 112, 148, 188, 203, 287, 307, 343, 358, 398, 428, 463, 482, 512, 538, 617, 643, 658, 727, 742, 812, 827, 853, 937, 967, 1007, 1022, 1028, 1058, 1072, 1112, 1253, 1267, 1282, 1288, 1358, 1373, 1442, 1457, 1462, 1483, 1513, 1553, 1567, 1568, 1618, 1637, 1652, 1813, 1828, 1847, 1897, 1912, 1982, 2003, 2008, 2023, 2092, 2107, 2113, 2177, 2183, 2198, 2267, 2282, 2393, 2437, 2443, 2458, 2477, 2528, 2542, 2612, 2638, 2653, 2722, 2723, 2737, 2738, 2807, 2822, 2828, 2848, 2932, 2983, 3002, 3023, 3037, 3067, 3158, 3178, 3262, 3268, 3277, 3283, 3353, 3368, 3437, 3452, 3478, 3548, 3563, 3613, 3632, 3647, 3697, 3752, 3808, 3823, 3892, 3893, 3907, 3947, 3977, 3983, 3998, 4003, 4087, 4102, 4178, 4192, 4193, 4207, 4388, 4402, 4432, 4438, 4453, 4493, 4523, 4577, 4607, 4633, 4648, 4718, 4753, 4802, 4817, 4907, 4922, 4948, 4978, 5032, 5062, 5102, 5117, 5153, 5257, 5312, 5348, 5362, 5363, 5377, 5468, 5543, 5557, 5572, 5578, 5608, 5648, 5663, 5732, 5747, 5803, 5818, 5858, 5908, 5923, 5942, 5972, 6007, 6077, 6103, 6118, 6187, 6202, 6272, 6287, 6293, 6313, 6397, 6467, 6488, 6518, 6532, 6572, 6623, 6727, 6733, 6742, 6748, 6818, 6832, 6833, 6902, 6917, 6943, 6973, 7013, 7027, 7028, 7078, 7097, 7112, 7118, 7162, 7273, 7288, 7357, 7372, 7442, 7448, 7463, 7468, 7547, 7552, 7567, 7573, 7643, 7658, 7727, 7742, 7853, 7897, 7903, 7918, 7937, 7987, 7988, 8002, 8042, 8072, 8098, 8113, 8182, 8183, 8197, 8267, 8273, 8282, 8288, 8392, 8443, 8483, 8497, 8527, 8548, 8618, 8702, 8722, 8728, 8743, 8813, 8828, 8897, 8912, 8938, 9008, 9043, 9073, 9092, 9107, 9157, 9197, 9212, 9268, 9283, 9352, 9367, 9407, 9437, 9443, 9458, 9472, 9547, 9638, 9652, 9653, 9667, 9703, 9758, 9862, 9898, 9913, 9953, 9983, 10037, 10067, 10093, 10108, 10198, 10213, 10262, 10297, 10367, 10382, 10408, 10438, 10492, 10522, 10562, 10577, 10583, 10613, 10627, 10808, 10822, 10823, 10837, 10913, 10928, 11012, 11017, 11032, 11038, 11068, 11108, 11122, 11123, 11192, 11207, 11263, 11318, 11368, 11383, 11402, 11452, 11467, 11537, 11563, 11578, 11647, 11662, 11732, 11738, 11747, 11753, 11837, 11857, 11948, 11978, 11992, 12013, 12032, 12083, 12167, 12187, 12193, 12208, 12277, 12278, 12292, 12293, 12362, 12377, 12403, 12473, 12487, 12538, 12557, 12572, 12578, 12622, 12733, 12748, 12817, 12832, 12838, 12902, 12908, 12923, 12992, 13007, 13012, 13033, 13103, 13118, 13168, 13187, 13202, 13363, 13378, 13397, 13447, 13448, 13462, 13502, 13532, 13553, 13558, 13573, 13642, 13657, 13727, 13733, 13748, 13762, 13903, 13943, 13957, 13987, 13993, 14008, 14048, 14078, 14162, 14188, 14203, 14273, 14288, 14357, 14372, 14398, 14477, 14503, 14533, 14552, 14587, 14617, 14657, 14672, 14708, 14728, 14812, 14827, 14867, 14903, 14917, 14918, 14932 }

Each of these solutions mod 15015 represents a unique combination of solutions mod 3, 5, 7, 11 and 13 (Cartesian product).
There is a number theoretical trick we can use to make findings these solutions mod 15015 a lot easier.

Lets say that we want to calculate the following combination for mod 15015:

$y = 1 \bmod 3$
$y = 3 \bmod 5$
$y = 1 \bmod 7$
$y = 5 \bmod 11$
$y = 5 \bmod 13$

For each solutions mod p, we divide and then multiple by every other prime. Using the following procedure:

For 1 mod 3 we get:

Step 1: $5^{-1} = 2 \bmod 3$ (calculate the inverse of 5 mod 3)
Step 2: $1 \times 2 \times 5 \bmod 3 \times 5$ (multiply the solution mod 3, 1 by the inverse and 5, and multiply the modulus by 5)
Step 3: $7^{-1} = 13 \bmod 15$ (calculate the inverse of 7 mod 15)
Step 4: $10 \times 13 \times 7 \bmod 15 \times 7$ (From step 2, carry over the 10 and multiply by the inverse and 7 and multiply the modulus by 7)
Step 5: $11^{-1} = 86 \bmod 105$ (calculate the inverse of 11 mod 105)
Step 6: $70 \times 86 \times 11 \bmod 105 \times 11$ (From Step 4 we get 70 mod 105 and we multiply that by the inverse and 11 and multiply the modulus by 11)
Step 7: $13^{-1} = 622 \bmod 1155$ (calculate the inverse of 13 mod 1155)
Step 8: $385 \times 622 \times 13 \bmod 1155 \times 13$ (From Step 6 we get 385 mod 1155 and we multiply that by the inverse and 13 and multiply the modulus by 13)

This gives us 5005 mod 15015

$5005 = 5 \times 7 \times 11 \times 13$ and $5005 \bmod 3 = 1$

Hence this simply lifts 1 mod 3 to mod 15015 by adding multiples of mod 5,7,11,13 while keeping it congruent to 1 mod 3.

There probably is a much more straight forward way to calculate the above, but it works and since it isn't bottle–necking my algorithms, that is all I care about.

Now to do the rest:

For 3 mod 5 we get:

Step 1: $3^{-1} = 2 \bmod 5$
Step 2: $3 \times 2 \times 3 \bmod 5 \times 3$
Step 3: $7^{-1} = 13 \bmod 15$
Step 4: $3 \times 13 \times 7 \bmod 15 \times 7$
Step 5: $11^{-1} = 86 \bmod 105$
Step 6: $63 \times 86 \times 11 \bmod 105 \times 11$
Step 7: $13^{-1} = 622 \bmod 1155$
Step 8: $693 \times 622 \times 13 \bmod 1155 \times 13$

This gives us 3003 mod 15015

For 1 mod 7 we get:

Step 1: $3^{-1} = 5 \bmod 7$
Step 2: $1 \times 5 \times 3 \bmod 7 \times 3$
Step 3: $5^{-1} = 17 \bmod 21$
Step 4: $15 \times 17 \times 5 \bmod 21 \times 5$
Step 5: $11^{-1} = 86 \bmod 105$
Step 6: $15 \times 86 \times 11 \bmod 105 \times 11$
Step 7: $13^{-1} = 622 \bmod 1155$
Step 8: $330 \times 622 \times 13 \bmod 1155 \times 13$

This give us 10725 mod 15015

For 5 mod 11 we get:

Step 1: $3^{-1} = 4 \bmod 11$
Step 2: $5 \times 4 \times 3 \bmod 11 \times 3$
Step 3: $5^{-1} = 20 \bmod 33$
Step 4: $27 \times 20 \times 5 \bmod 33 \times 5$
Step 5: $7^{-1} = 118 \bmod 165$
Step 6: $60 \times 118 \times 7 \bmod 165 \times 7$
Step 7: $13^{-1} = 622 \bmod 1155$
Step 8: $1050 \times 622 \times 13 \bmod 1155 \times 13$

This gives us 6825 mod 15015

For 5 mod 13 we get:

Step 1: $3^{-1} = 9 \bmod 13$
Step 2: $5 \times 9 \times 3 \bmod 13 \times 3$
Step 3: $5^{-1} = 8 \bmod 39$
Step 4: $18 \times 8 \times 5 \bmod 39 \times 5$
Step 5: $7^{-1} = 28 \bmod 195$
Step 6: $135 \times 28 \times 7 \bmod 195 \times 7$
Step 7: $11^{-1} = 1241 \bmod 1365$

Step 8: 525 × 1241 × 11 mod 15015

This gives us 4620 mod 15015

Adding the results for mod 3,5,7,11,13 together: 5005 + 3003 + 10725 + 6825 + 4620 = 148 mod 15015

In my original paper I would call these intermediate results, partial results. I think that is a fitting name. When we sum up these partial results, we get 148 mod 15015. This way we can reduce finding combinations modulo a composite number to summing together partial results constructed from the prime factors of the composite modulus.

Now what we could do is, calculate all the possible coefficient solutions mod 3,5,7,11,13 and then use the above calculations to create partial results from them mod 15015:

Before:

y mod 3 = { 1, 2 }
y mod 5 = { 2, 3 }
y mod 7 = { 0, 1, 6 }
y mod 11 = { 1, 2, 5, 6, 9, 10 }
y mod 13 = { 1, 5, 6, 7, 8, 12 }

After:

y mod 3 = { 5005, 10010 }
y mod 5 = { 12012, 3003 }
y mod 7 = { 0, 10725, 4290 }
y mod 11 = { 1365, 2730, 6825, 8190, 12285, 13650 }
y mod 13 = { 6930, 4620, 11550, 3465, 10395, 8085 }

One of my original research approaches was the insight that if we could growing the modulus by adding more and more primes, eventually $p + q$ will end up being the smallest solution mod m. The other solutions will keep growing to a number bigger then N.
So my initial idea is, if we generate these partial results, and we select one partial result from each prime modulus and sum them together mod m, how do we find the smallest sum mod m? This however is a modular multiple-choice subset-sum problem. Not easily solve-able.
In my original paper I also discussed constructing partial results in mod m_0 and m_1 each from unique primes and as long as m_0 and m_1 are large enough, then we can use the intersection between both sets of solutions to further narrow down the possible set of solutions.

For example:

Mod m_0 = 3 × 7 × 13 × 19 (5187)

y mod 3 = { 1729, 3458 }
y mod 7 = { 0, 4446, 741 }
y mod 13 = { 1197, 798, 1995, 3192, 4389, 3990 }
y mod 19 = { 3003, 3822, 1638, 2457, 273, 4914, 2730, 3549, 1365, 2184 }

Mod m_1 = 5 × 11 × 17 × 23 (21505)

y mod 5 = { 8602, 12903 }
y mod 11 = { 13685, 5865, 3910, 17595, 15640, 7820 }
y mod 17 = { 0, 12650, 10120, 16445, 7590, 13915, 5060, 11385, 8855 }
y mod 23 = { 0, 18700, 7480, 1870, 17765, 14960, 6545, 3740, 19635, 14025, 2805 }

From m_0 we select: 1729 + 4446 + 798 + 3549 = 148 mod 5187 and from m_1 we select: 12903 + 3910 + 11385 + 14960 = 148 mod 21505
As you can see, 148 ($p + q$) can be found as a sum mod m_i , this will hold true for any modulus bigger then $p + q$. Hence by inspecting intersections between solution sets, we can quickly narrow it down to one single solution. But in practice, since we need a modulus bigger then $p + q$, the amount of possible sums, aka the order of the Cartesian product, quickly grows. Hence this is not feasible, but nonetheless, it is an interesting direction to approach this problem.
In my first paper I attempted to find these intersections using the LLL algorithm[9]. However many improvement can be made there, and would I write it today, there would be many things I would change and simplify further. This earlier work (do note I had only been teaching myself math for about a year at this point) can be found here:

https://github.com/BigPolarBear1/factorization_v1

V. Quadratic sieve and beyond

After my attempt at finding intersections between solutions sets in mod m_i using LLL, I instead used these findings to generate smooth candidates for the Quadratic Sieve algorithm. I will now quickly describe the transformations I used to achieve this.

Let us factor 4387.

We set the factor base b to:

b = { 3, 5, 7, 11, 13 }

Next using the quadratic formula: $x^2 + yx + N$ mod b_i we calculate all the possible coefficient solutions for each prime in the factor base:

y mod 3 = { 1, 2 }
y mod 5 = { 2, 3 }
y mod 7 = { 0, 1, 6 }
y mod 11 = { 1, 2, 5, 6, 9, 10 }
y mod 13 = { 1, 5, 6, 7, 8, 12 }

Next we create a hashmap and go over each coefficient and calculate the following linear congruence:

$x \times N = y^2 \bmod b_i$

For coefficient solution $y = 5 \bmod 11$ we would get:

$x \times 4387 = 5^2 \bmod 11$
$\Rightarrow x = 4$

We save the x solution as key in the hashmap and the coefficient solution as value:

mod 3 = 1 : { 1, 2 }
mod 5 = 2 : { 2, 3 }
mod 7 = 0 : { 0 }, 3 : { 1, 6 }
mod 11 = 5 : { 1, 10 }, 9 : { 2, 9 }, 4 : { 5, 6 }
mod 13 = 11 : { 1, 12 }, 2 : { 5, 8 }, 6 : { 6, 7 }

Next we iterate sieve interval i from 0 to i_{n-1}

For example when $i = 97$ we check if 97 mod 3, 97 mod 5, .., is a key in the hashmap and we collect the results.

97 mod 3 = 1 : { 1, 2 }
97 mod 5 = 2 : { 2, 3 }
97 mod 7 = /
97 mod 11 = 9 : { 2, 9 }
97 mod 13 = 6 : { 6, 7 }

At $i = 97$ we found results in 4 out 5 elements in the coefficient solution set for factor base b.
We multiply the moduli together for which we found a result:

$3 \times 5 \times 11 \times 13 = 2145$.

And if this is bigger then $\sqrt{i \times N}$ we continue.

Next we calculate the partial results for the coefficient solutions we just collected (Chapter IV):

y mod 3 = { 715, 1430 }
y mod 5 = { 1287, 858 }
y mod 11 = { 585, 1560 }
y mod 13 = { 825, 1320 }

Next we generate combinations mod 2145 choosing at most one partial result per modulus.

For example: 715 + 1287 + 585 + 825 = 1267 mod 2145

Lets call this the coefficient candidate y.

The useful thing about this setup is that if we now calculate $y^2 - i \times N$ we know the result will be divisible by the moduli from which we collected the partial results.

Hence $1267^2 - 97 \times 4387 = 1179750$ $(2 \times 3 \times 5^3 \times 11^2 \times 13)$

All but one of the factors are in the factor base (2) in this example.

The closer y^2 is to $i \times N$, the smaller the smooth candidate will be.
Once enough such smooth candidates are found, you finish the rest of the algorithm using the default Quadratic Sieve proceedings.
Which I won't reiterate as this is widely documented. But in short you would use Gaussian Elimination[10] or Block Lanczos[11] to find a combination of smooths that can be multiplied together to form a square relation on both sides of the congruence mod N.

You can find the code for my python implementation here: https://github.com/BigPolarBear1/factorization_v2

VI. Quadratic coefficients

In the above, we generate possible linear coefficients y and then square them. By subtracting $i \times N$, which I shall henceforth refer to as simply iN, we can then predict at-least some of the factors of the smooth candidates.

Let us explore how this iN value relates to the coefficient of the quadratic term, an important subject we have not touched on yet.

For example if we have $y_0 = 148$ we see that $148^2 - 4 \times 4387 = 66^2$ ($i = 4$ thus $iN = 4 \times 4387$) satisfies our square relation mod N.
Also note that $148^2 - 4 \times 4387 = 66^2$ is the formula for the quadratic discriminant, which makes sense.
We can see where $i = 4$ comes from when we subtract and add both linear coefficients from each-other and look at the factorization of the result:

148 – 66 = 82 (41 × 2)
148 + 66 = 214 (107 × 2)

When subtracting we get 2 times the lower factor and when adding we get 2 times the upper factor.
We multiply the factors we found, excluding the factors of N and we get: $2 \times 2 = 4$.

Another example when $y_1 = 3$ and $y_0 = 1602$:

$1602^2 = 3^2 \bmod N$
$1602 - 3 = 1599\ (41 \times 39)$
$1602 + 3 = 1605\ (107 \times 15)$

Hence we get $i = 39 \times 15 = 585$ and verifying this:

$1602^2 - 4387 \times 585 = 3^2$

And when $y_1 = 1$ and $y_0 = 534$:

$1^2 = 534^2 \bmod N$
$534 - 1 = 41 \times 13$
$534 + 1 = 107 \times 5$

Hence we get $i = 13 \times 5 = 65$ and verifying this:

$534^2 - 4387 \times 65 = 1^2$

This i value is actually the coefficient for the quadratic term (you may recognize this in the discriminant formula).
In the example that $y_1 = 1$ and $y_0 = 534$ we have a i value of 13×5.
We know from the above explanation in the first chapters that the roots represent the factors of N.

We can see that the following holds:

$13 \times (-41)^2 - 1 \times (-41) + 41 = 0 \bmod 4387$
$5 \times (-107)^2 + 1 \times (-107) - 107 = 0 \bmod 4387$

I will define the quadratic coefficient as z.

However, in the above example we note that if we have an odd linear coefficient we get quadratics of the shape: $zx^2 + yx + x$.
Because the quadratic for even coefficients is simpler ($zx^2 + yx$) we shall restrict ourselves to these alone.

Thus working with even coefficient if we have $y_1 = 2$ and $y_0 = 1068$, we see that:

$1068 - 2 = 26 \times 41$
$1068 + 2 = 10 \times 107$

Since we are now working with two even coefficients we get a quadratic of the shape: $zx^2 + yx$

This means the quadratic coefficients become 26/2 and 10/2 (since when we take the derivative, it gets multiplied by 2 from the quadratic exponent).

$13 \times (-41)^2 - 2 \times (-41) = 5 \times 4387$ or $0 \bmod 4387$
$5 \times (-107)^2 + 2 \times (-107) = 13 \times 4387$ or $0 \bmod 4387$

And the derivative reveals the other linear coefficient:

$26 \times (-41) - 2 = -1068$

And the quadratic for $y_0 = 1068$:

$13 \times (-41)^2 + 1068 \times (-41) = -5 \times 4387$ or $0 \bmod 4387$
$5 \times (-107)^2 + 1068 \times (-107) = -13 \times 4387$ or $0 \bmod 4387$

Our quadratic with the addition of quadratic coefficients is now: $zx^2 + y_0x = zx^2 + y_1x \bmod N$

For my next attempt (factorization_v3) I realized that what I was doing in factorization_v2 was basically multiple-polynomial quadratic sieve, without the sieve interval. I should note that MPQS has wildly different polynomials, and their way of representing their quadratics has nothing to do with how we approach the problem. What we are doing is wildly different but allowed us to come to the same conclusions. The most critical step now is figuring out how we can gain an advantage with our approach. I am considering the following:

For factorization v3 I added a sieve interval, similar to SIQS. Another thing that we can do since we can calculate all this information at a low cost is creating a 2d sieve interval. Where the columns represent the linear coefficient and the rows the quadratic coefficient. For the sieve interval in the direction of the linear coefficient you need to add the modulus at each step. For the quadratic coefficient we simply find quadratic coefficients z where $N \times 4 \times z$ is a quadratic residue of the modulus. For version 3, I am currently exploring two approaches. The big one is finding a bridge to number field sieve. By using quadratic coefficients, a trick we can do now is to pull all our smooths from a single modulus (since we can check the same modulus at different quadratic coefficients). We then simply find smooths mod m and after the linear algebra step take the square root over a finite field. This is of-course a wild over simplification, but I believe it may be possible and am exploring this path. Should that fail, what we can definitely do is targeted smooth finding. After factoring a smooth, we can go look for factors which turned up with odd exponents in different quadratic coefficients. Since this is something we can efficiently do now. This should wildly lower the amount of smooths needed to achieve successful factorization of N.

And finally, we can now also achieve p-adic lifting at very low computational cost. Since both sides of the congruence must yield the same result mod p_i^a this allows us to reduce the amount of possible solutions. Thus for any given linear coefficient and quadratic coefficient there should only be one valid solution when lifted to higher exponents. We can use this to further reduce the amount of required smooths by working with smaller factor bases.

You can find factorization_v3 here (I'm still writing my 2d sieving implementation at the time of writing and will experiment if lifting boosts performance also):

https://github.com/BigPolarBear1/factorization_v3

VII. Looking forward and building toward factorization v4

Having worked out all this number theory, we can create a hashmap which we can key by either linear coefficient or quadratic coefficient.

For example if $N = 4387$ we would get the following hashmap for primes 3,5,7:

y_1 mod 3:

1 : { 0, 2 }
2 : { 0, 2 }
0 : { 0, 1 }

y_1 mod 5:

2 : { 0, 2, 4 }
3 : { 0, 2, 4 }
0 : { 0, 2, 3 }
1 : { 0, 1, 3 }
4 : { 0, 1, 3 }

y_1 mod 7

0 : { 0, 3, 5, 6 }
3 : { 0, 1, 2, 5 }
4 : { 0, 1, 2, 5 }
2 : { 0, 2, 3, 4 }
5 : { 0, 2, 3, 4 }
1 : { 0, 1, 4, 6 }
6 : { 0, 1, 4, 6 }

Here we key our hashmap by linear coefficient y_1 and display the set of possible quadratic coefficients mod p_i that can occur.

For example at y_1 mod 7 if we have linear coefficient 2 we get the set of quadratic coefficients: { 0, 2, 3, 4 }

Which gives us the following 4 congruences, for each of which when we calculate the discriminant formula, we should get another quadratic residue mod 7 (do note this looks different for odd linear coefficients, see previous chapter):

$0x^2 + 2x - 4387 = 0 \bmod 7 \Rightarrow 2^2 + 4387 \times 4 \times 0 = 4$
$2x^2 + 2x - 4387 = 0 \bmod 7 \Rightarrow 2^2 + 4387 \times 4 \times 2 = 2$
$3x^2 + 2x - 4387 = 0 \bmod 7 \Rightarrow 2^2 + 4387 \times 4 \times 3 = 1$
$4x^2 + 2x - 4387 = 0 \bmod 7 \Rightarrow 2^2 + 4387 \times 4 \times 4 = 0$

The other quadratic coefficients will yield non-square residues, hence those are invalid solutions.

The idea I want to explore in factorization v4 is the following.
For any linear coefficient pairing y_0 and y_1 mod m we can also calculate all the possible quadratic coefficients. Each of these possible quadratic coefficient mod m will eventually yield a solution that is a square relation in the integers when calculating the discriminant formula, the problem is simply how often the modulus needs to be added. For some quadratic coefficients this can be a small number, for others an enormous number. This problem however of figuring out how many times the modulus needs to be added, I want to attempt to solve it with a lattice basis reduction algorithm, as I think this is a good candidate for it. Additionally, you may be able to somehow sieve this value by looking at primes outside the modulus since if it is the correct quadratic coefficient, the result is a square in the integers, or every possible mod p.

You can track my work in progress for v4 here: https://github.com/BigPolarBear1/factorization_v4

VIII. Conclusion

We have managed to make many reductions in complexity in the factorization problem. Where traditionally modern variants of Fermat's factorization method, which includes Number Field sieve and Quadratic sieve, had to find smooth numbers within a factor base, to then hopefully complete a square relation mod N. We have now found a way to instead enumerate the roots of the squares in this square relation by calculating coefficients of the linear term of a quadratic mod p_i. This is done in an attempt to break the almost 40-year stalemate in factorization algorithms. I would now encourage the reader, to continue this work, as this is merely the beginning, and many unknown lands of modular magic and polynomials lay ahead of us to explore.

IX. References

1. Number Field Sieve: https://en.wikipedia.org/wiki/General_number_field_sieve
2. RSA algorithm: https://en.wikipedia.org/wiki/RSA_cryptosystem
3. Euclidean algorithm: https://en.wikipedia.org/wiki/Euclidean_algorithm
4. Fermat's factorization method: https://en.wikipedia.org/wiki/Fermat's_factorization_method
5. Quadratic sieve: https://en.wikipedia.org/wiki/Quadratic_sieve
6. Carl Pomerance: https://en.wikipedia.org/wiki/Carl_Pomerance
7. Subset-sum problem: https://en.wikipedia.org/wiki/Subset_sum_problem
8. Legendre Symbol: https://en.wikipedia.org/wiki/Legendre_symbol
9. LLL algorithm: https://en.wikipedia.org/wiki/Lenstra–Lenstra–Lovász_lattice_basis_reduction_algorithm
10. Gaussian Elimination: https://en.wikipedia.org/wiki/Gaussian_elimination
11. Block Lanczos: https://en.wikipedia.org/wiki/Block_Lanczos_algorithm