Replication of Quantum Factorisation Records with an 8-bit Home Computer, an Abacus, and a Dog

Peter Gutmann, University of Auckland   Stephan Neuhaus, Zürcher Hochschule für Angewandte Wissenschaften

March 2025

1 Abstract

This paper presents implementations that match and, where possible, exceed current quantum factorisation records using a VIC-20 8-bit home computer from 1981, an abacus, and a dog. We hope that this work will inspire future efforts to match any further quantum factorisation records, should they arise.

2 Introduction

In 1994, mathematician Peter Shor proposed his quantum factorisation algorithm[1], now known as Shor's Algorithm [1]. In 2001, a group at IBM used it to factorise the number 15 [2]. Eleven years later this was extended to factorise the number 21 [3]. Another seven years later a factorisation of 35 was attempted but failed [4]. Since then no new records have been set[2], although a number of announcements of such feats have cropped up from time to time alongside the more publicly-visible announcements of quantum supremacy every few months. These announcements are accompanied by ongoing debates over whether a factorisation actually took place and if so what it was that was factorised [5][6], with the issue covered in more detail in section 3. Of particular note was the claim in 2024 by researchers to have factorised an RSA-2048 number [7][8] ("the D-Wave paper"). In this paper we focus on the factorisations of 15, 21, and 35, as well as the claimed RSA-2048 factorisation[3].

2.1 Terminology

New technologies, when introduced, are typically given names that overstate their capabilities, usually by equating them with existing familiar systems or technological artefacts. For example the first computers in the 1940s and 1950s, often little more than glorified electric adding machines, were nevertheless described as "electronic brains". More recently, large language models (LLMs) have been touted as "artificial intelligence", and complex physics experiments have been touted as "quantum computers". In order to avoid any confusion with

[1] We use the UK form "factorise" here in place of the US variants "factorize" or "factor" in order to avoid the 40% tariff on the US term.

[2] It should be pointed out that the above papers never claimed to be setting any factorisation records but merely provided an experimental realisation of Shor's Algorithm. The claims were added later by the media.

[3] We note that the paper has, at the time of writing (March 2025) a publication date in the future (June 2025). It appears that the D-Wave device can also shift time and relative dimensions in space.

actual computers like the VIC-20 with which they have nothing in common, we refer to them here as “physics experiments”. Similarly, we refer to an abacus as “an abacus” rather than a digital computer, despite the fact that it relies on digital manipulation to effect its computations. Finally, we refer to a dog as “a dog” because even the most strenuous mental gymnastics can’t really make it sound like it’s a computer.

3 Quantum Factorisation Overview

When stage magicians perform sleight-of-hand tricks, traditionally card tricks, they use specially constructed decks called force decks with which they can force the participants in the trick to pick a card of the magician’s choosing. An example of such a force deck is a Svengali deck which when shown to the participant or the audience appears to contain a standard mix of cards but which only contains a single repeated and therefore entirely predictable card.

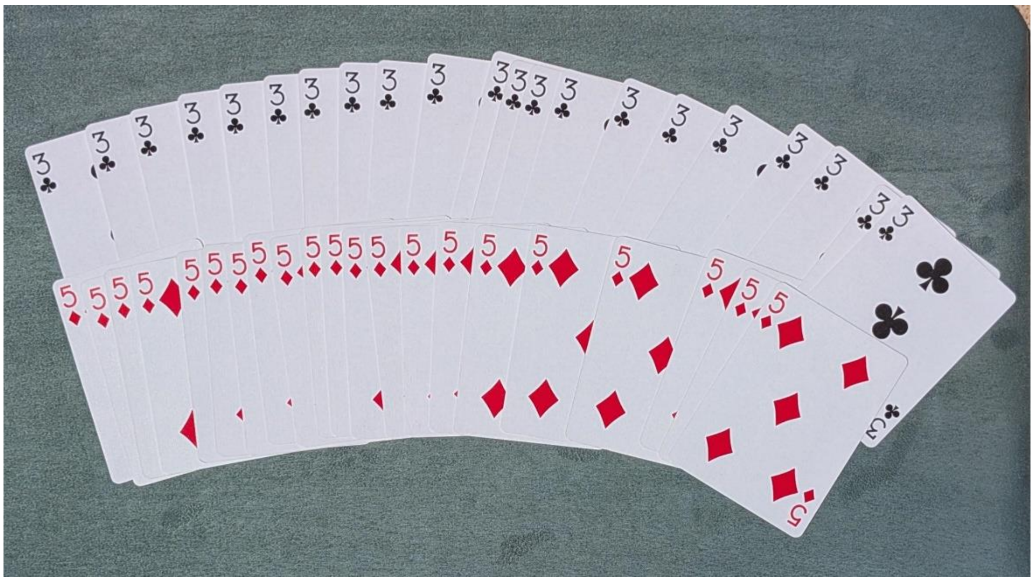

Figure 1: Force deck being used to replicate the quantum factorisation of 15. “Is one of the cards a three?”

Similarly, quantum factorisation is performed using sleight-of-hand numbers that have been selected to make them very easy to factorise using a physics experiment and, by extension, a VIC-20, an abacus, and a dog. A standard technique is to ensure that the factors differ by only a few bits that can then be found using a simple search-based approach that has nothing to do with factorisation. For example the RSA-2048 number
2344221089529646655151068154361983197810258179973661124697652159019189322413
5789025070678051976867349306593332331728775086731364111282889875974451560408
7401460159349869904762142706400868174255815381703738702593130665837689036970
4828064146736741158993910041461135601151339797803821866970974724786872772467
6001584905770525234976669382895464232871732123454572174833964467804115311936
8505867914928449735609052294298924389262041881744905437550809726216528316509
3027743111302874592959317102563951824995592125577639307824751973466650905577
6152948501360345202224227559964438653352949732541506721438058592990053089448
078211591 is the product of two factors

1531084938705115293431839826945810375548166939018931860902798006004492850911
0927257807106642733607032169360156227443309858061960009966390541027902314815
2523939650071615596077413516469321466486454921404568342497216591961439354064
8442582007387324342415272089894881983294008201158253359215854823896119936678
49543 and
1531084938705115293431839826945810375548166939018931860902798006004492850911
0927257807106642733607032169360156227443309858061960009966390541027902314815
2523939650071615596077413516469321466486454921404568342497216591961439354064
8442582007387324342415272089894881983294008201158253359215854823896119936678
49537 that differ by only one or two bits, making it possible to perform the "factorisation" through a simple integer square root calculation. Note that such a value would never be encountered in the real world since the RSA key generation process typically requires that |p-q| > 100 or more bits [9]. As one analysis puts it, "Instead of waiting for the hardware to improve by yet further orders of magnitude, researchers began inventing better and better tricks for factoring numbers by exploiting their hidden structure" [10].

A second technique used in quantum factorisation is to use preprocessing on a computer to transform the value being factorised into an entirely different form or even a different problem to solve which is then amenable to being solved via a physics experiment. For example the 2019 quantum factorisation of 1,099,551,473,989 [11], of which more below, relied on processing with a computer to transform the problem into one that was solvable with a three-qubit circuit [12].

Other quantum factorisations also rely on computers to reduce the problem to a form in which it can be "solved" through a physics experiment. Even the factorisations of 15 and 21 used the so-called compiled form of Shor's algorithm which uses prior knowledge of the answer to merely verify the (known-in-advance) factors rather than performing any actual factorisation. In the case of 15 and 21 it reduced the number of qubits required from 8 and 10 to 2 in the compiled form. The paper that discusses this result comments that "it is not legitimate for a compiler to know the answer to the problem being solved. To even call such a procedure compilation is an abuse of language" [13].

The paper then presents the factorisation of a 768-bit (231-digit) number and a 20,000-digit number, both of which can also be factorised using 2 qubits in the compiled form. As the paper points out, "our technique can factor all products of p, q such that p, q are unequal primes greater than two, runs in constant time, and requires only two coherent qubits" [13].

Still other quantum factorisations go even further. For example one claimed factorisation involved working backwards from the known answer to design a physis experiment that produced the known-in-advance solution. There is no equivalent computation for such a sleight-of-hand operation so we have no means to show an equivalent using a VIC-20 or an abacus. The trick in all of these cases is to figure out how to construct a value such that it can then be transformed into a vastly simpler form in which it can be "factorised" via a physics experiment.

These types of factorisations have also been referred to as "stunt factorisations" [14][15]. For example the main effort in the 2012 factorisation of 143 into 11 × 13 [16] consisted of finding a value with the special properties required that allowed it to be “factorised” by a physics experiment. This feat was then extended in 2014 to 56,153 [17], in 2018 to 4,088,459 [18] and later that year to the impressive-looking 383,123,885,216,472,214,589,586,724,601,136,274,484,797,633,168,671,371 [19].

Many further types and techniques for stunt factorisations exist, far too many to catalogue here, with the practice typically being to manufacture a small value that’s easily “factorised” via a physics experiment (143) and then later figuring out how to stretch the value to add more and more digits (383,123,885,216,472,214,589,586,724,601,136,274,484,797,633,168,671,371) while still allowing it to be “factorised” by the same physics experiment [15]. For example the compiled Shor’s algorithm can factorise any composite number $p \times q$ on a very small physics experiment [13], a “factorisation” mechanism that has been given the tongue-in-cheek name Smolin-Smith-Vargo Algorithm after the authors of the paper that pointed out the technique [10]. It should be noted here that all of these sleight-of-hand and stunt values are trivially factorised by Fermat’s method on a Raspberry Pi or similar.

Similar to stage magic, the exercise when responding to a new quantum factorisation announcement is not only to marvel at the trick but to try and figure out where the sleight-of-hand occurred. One simple technique to catch the use of sleight-of-hand numbers is to view them in binary form. If they consist almost entirely of zero bits, as did the 2019 factorisation of 1,099,551,473,989, which begins with 100000000000000… when expressed in binary, then it’s a sleight-of-hand number. Similarly, numbers with repeating bit patterns 10101010… or similar are sleight-of-hand numbers. Section 7 presents a technique for selecting non-sleight-of-hand numbers for future quantum factorisation work.

A second technique is to check whether the value submitted to the physics experiment was the one being factorised or whether it has been first transformed on a computer into an entirely different form that’s solvable with a physics experiment. A standard trick here is to transform the factorisation into a combinatorial minimization problem which is readily solved using Grover’s algorithm [20], completely impractical for factorisation but perfectly suitable for publication credit.

Many other sleight-of-hand tricks exist for creating apparent quantum factorisations. One example is what we are calling the Callas Normal Form for Sleight-of-Hand Quantum Factorisation or “Callas Normal Form” for short after cryptographer Jon Callas who first described it [21]. In the Callas Normal Form, the factors are integers $p = 2^{n-1}$ and $q = 2^{m+1}$, where $n \leq m$, and p and q are ideally prime, but don’t have to be. The binary representation of the product $N = pq$ then starts with n one bits followed by $m - n$ zero bits and ends in n one bits. Needless to say, this is easily detected, even on a 6502, and easily factorised (no real-world RSA toolkit would ever generate such prime factors). For example, a recent preprint [22] uses this form to claim in its title success in factorising 4096-bit integers with Shor’s algorithm “under certain conditions”, where the “conditions” for the 12 examples used turn out to be equivalent to the Callas Normal Form.

So far as we have been able to determine, no quantum factorisation has ever factorised a value that wasn’t either a carefully-constructed sleight-of-hand number or for which most of the work wasn’t done beforehand with a computer in order to transform the problem into a different one that could then be readily solved by a physics experiment [23][10]. We attempt to address this deficiency by providing criteria for evaluating quantum factorisation attempts in section 7.

The pervasive use of sleight-of-hand numbers and techniques and stunt factorisations throughout the field of quantum factorisation makes it difficult to select targets for our factorisation replication attempts. Since it's possible, with a bit of thought, to construct arbitrarily impressive-looking values for factorisation, an example being the 20,000-digit artificial value that was factorised with a 2-qubit physics experiment [13], we have to select targets that are at least within shouting distance of an actual application of something like Shor's algorithm for quantum factorisation. The three instances of this that we have been able to identify in the literature, even though they also use sleight-of-hand by using the compiled form of Shor's algorithm mentioned earlier, are the 2001 factorisation of 15, the 2012 factorisation of 21, and the (attempted) 2019 factorisation of 35, constituting not actual quantum factorisations but at least the least sleight-of-handy attempts at quantum factorisation.

4 Performing Quantum Factorisation operations with a VIC-20

The VIC-20 was a popular home computer in the 1980s that used the then popular 6502 microprocessor from 1975. Since this processor uses transistors, and transistors work by using quantum effects, a 6502 is as much a quantum device as is a D-Wave **"**quantum computer**"**. The 6502 has three 8-bit registers, an 8-bit data bus, and a 16-bit address bus, giving it a 64 KiB address space. The VIC-20 had 20 KiB ROM and 5 KiB RAM, with 3.5 KiB available for applications, and features a roughly 1 MHz clock.

The processor has neither multiply nor divide instructions, so if we want to factor the 2001, 2012 and 2019 values, we have to resort to a different technique. The easiest way to factor a number with factors less than 16 is probably to pre-compute a 16 × 16 multiplication table so that the entry at row i column j is i × j, for $0 \le i, j < 16$. When factoring a number $x \le 225$, we would go through the table by indices (0,0), (1,0), (2,0), (1,1), (3,0), (2,1), (4,0), (3,1), (2,2) and so on until we find the number to be factored, in which case the smallest of the indices is either 1 (so that x is itself prime) or is the smallest prime factor of x. If we do not find the number, it must be prime (because a composite integer at most 225 must have a prime factor of at most sqrt(225) = 15).

For example, to factor 15, we would go through the table and stop with 15 = 5 × 3 as the answer. This method will successfully factor the 2001, 2012, and 2019 values because they all had factors that were less than 16. There are obvious optimisations and extensions, such as keeping only half of the table, removing rows and columns where one of the factors is 0 or 1 and thus extending the range of the table, but that would complicate the program, and the technique stays essentially the same.

The RSA-2048 moduli factored in the D-Wave paper are so large that this technique cannot be used on them, but are also sleight-of-hand numbers: the moduli were been specially chosen so that their prime factors p and q are either 2 or 6 apart. That means that for each of these RSA-2048 moduli, there exists a 1024-bit integer x that lies midway between p and q, and an integer d such that $p = x - d$ and $q = x + d$, where d is either 1 (if p and q are 2 apart) or 3 (if they are 6 apart). This means that there is just one bit of uncertainty in the whole factorisation, namely whether d is 1 or 3.

The key to making this work on a VIC-20 is the realisation that $N = pq = (x - d)(x + d) = x^2 - d^2$, or in other words, $N + d^2 = x^2$. That means that $N + d^2$ is a perfect square and d is either 1 or 3. In order to factor N, we can thus follow the procedure given in Figure 2.

1. Compute the integer square root x of N + 9 with remainder.
2. If the remainder is zero, output $p = x - 3$ and $q = x + 3$. Stop.
3. If the remainder is 8, output $p = x - 1$ and $q = x + 1$. Stop.
4. Output "Unfair!" because p and q will not be 2 or 6 apart and therefore aren't in the special form required for them to be "factored" using a physics experiment. Stop.

Figure 2: VIC-20 RSA-2048 factorisation algorithm

The problem is now to compute an integer square root with remainder. Luckily, John von Neumann adapted an algorithm apparently created for use with an abacus to the EDVAC in 1945 [24]. This was then translated by Henry S. Warren, Jr. into modern notation [25], with Figure 3 showing the "hardware algorithm" on p. 210, slightly modified to replace the original bit mask m with the bit number to set.

```
1 unsigned isqrt(unsigned x) {
2 unsigned m, y, b;
3
4 m = 30;
5 y = 0;
6 while (m >= 0) {            // Do 16 times.
7     b = y | (1 << m);
8     y = y >> 1;
9     if (x >= b) {
10        x = x - b;
11        y = y | (1 << m);
12     }
13     m = m - 2;
14   }
15   return y;
16 }
```

Figure 3: Integer square root algorithm

This computes the integer square root of x in y and leaves the remainder (i.e., $x - y \times y$ for the original value of x) in x. To implement this for a VIC-20 and for numbers of the size for RSA-2048, what is an "unsigned" in Warren's notation becomes an array of 256 bytes ("bignum"). Then the only operations one needs to implement on bignums are the ones shown in Figure 4.

1. Zeroing a bignum (y = 0 in line 5)
2. Copying a bignum to another (b = y | m in line 7)
3. Shifting a bignum right by 1 (y = y >> 1 in line 8)
4. Checking whether one bignum is greater than or equal to another (if (x >= b) in line 9)

5. Subtracting two bignums (x = x – b in line 10)
6. Checking whether a bignum is equal to a small integer (to check if the remainder is 0 or 8)
7. Set a bit in a bignum (b = y | (1 << m) in line 7, and y = y | (1 << m) in line 11).

Figure 4: VIC-20 "bignum" operations

All of these operations are straightforward to implement for bignums on a 6502. Note especially that neither division or multiplication of bignums is needed.

The assembler code [26] was then compiled and linked with the cc65 cross-compiler suite, written and maintained by a large number of 6502 enthusiasts [27]. The source code consists of 427 lines (without comments or empty lines). The assembled and linked ROM image contains 794 code and data bytes, 256 of which are for the modulus to be factored. That leaves only 538 bytes for the code proper. It needs 1792 bytes of RAM, all of which fits easily into the mighty VIC-20 (3.5 KiB usable RAM, 20 KiB ROM). The code was not particularly optimised, either for space or for time.

The code is designed to be ROMed in a 20 KiB EEPROM, to be put at address $b000. The code was run on Mike Chamber's excellent cycle-accurate fake6502 emulator [28]. This is a one-source-code-file-fits-all emulator that is simply made to be used in other projects.

The D-Wave paper featured ten RSA-2048 moduli, which we number from 0 to 9. Our code factored all of them correctly in the times given in Table 1.

Modulus	Ticks
0	16609726
1	16352636
2	16704327
3	16281246
4	16422636
5	16321994
6	16367815
7	16549115
8	16188092
9	16446609

Table 1: VIC-20 Factorisation cycle times

Ticks are counted from executing the first instruction in the reset routine loaded from address $fffd to finishing the main function. With the VIC-20's 1 MHz clock, one factorisation will thus take roughly 16.5 seconds. Running the code on the emulator on a ThinkPad X1 took less than one second, including loading the modulus from an input file and writing the factors to output files.

To summarise then, a VIC-20 can easily factor all the integers that have currently been factored with Shor's algorithm (the 2001 and 2012 values 15 and 21), a number that was slightly too large for Shor's Algorithm on current physics experiments (the 2019 value 35), and all of the

2048-bit RSA moduli in the D-Wave paper. The factorisations were efficient both in terms of memory use and execution time.

We have thus broken, or at least also replicated, all quantum factoring records, and have additionally replicated a 2025 result with 1981 technology using an algorithm for a 1945 computer.

5 Performing Quantum Factorisation operations with an Abacus

As section 4 pointed out, the "factorisation" algorithm used for the RSA-2048 values was apparently originally created for use with an abacus [29]. This means that we can factorise both the 2001 and 2012 values, the 2019 attempt, and the RSA-2048 value using an abacus. Factorising the 2001, 2012 and 2019 values is trivially performed through trial multiplication, which is slightly easier than trial division when performed on an abacus. However the multiplication process is complicated by the fact that multiplying multi-digit values on an abacus is performed by multiplying a digit at a time, which is done mentally. Since all three of the values 15, 21, and 35 have single-digit multiplicands, there's nothing for the abacus to do.

Therefore in order to be able to use an abacus for our factorisation we need to use division, in this case division by three for the successful factorisations and division by five for the unsuccessful one. Division on an abacus begins at the leftmost (most significant) digit, in the case of 15 a 1.

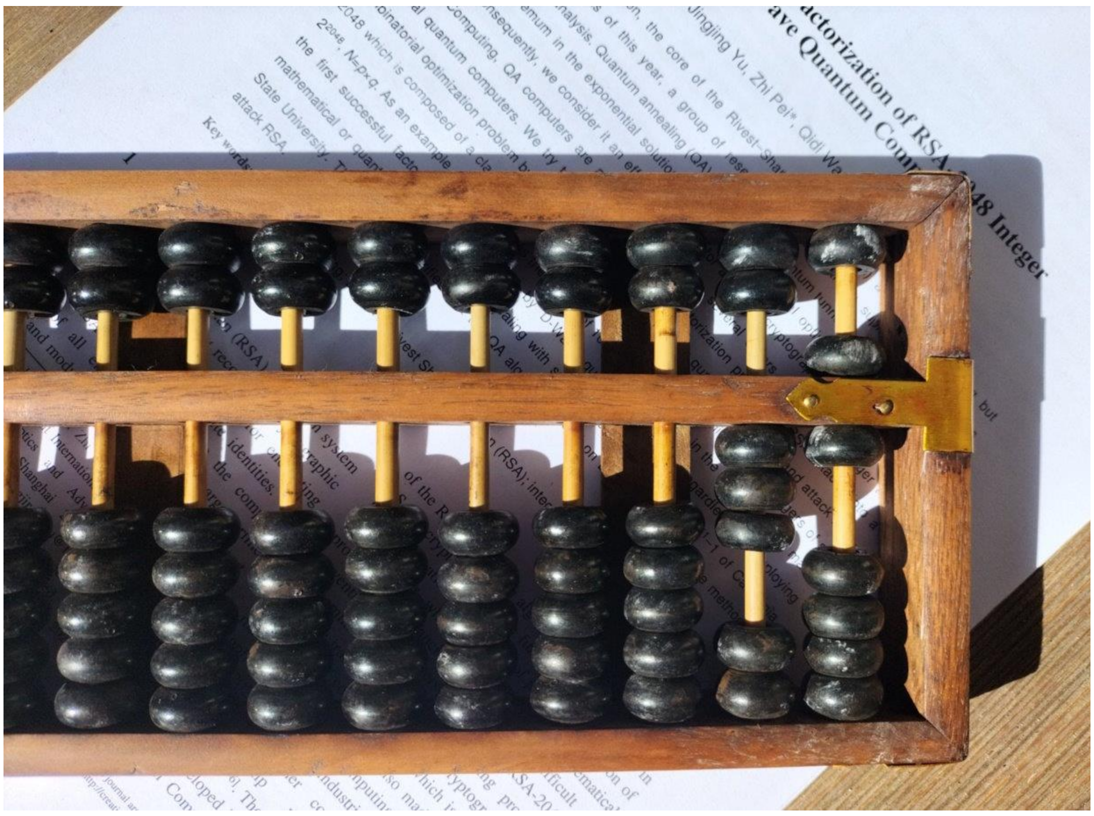

Figure 5: An abacus in the process of replicating the quantum factorisation of 15

The rule for dividing a one digit (in the tens column) is "one by three is three plus one", so our ten becomes a three with the remainder added to the next column along [30]. We now have the value 36 as shown in Figure 5, and move on to the next digit, 6. The rule for this is "cancel the six, forward two", which means clear the value 6 and add two to the column to the left, which is now 5.

So the result of dividing 15 by 3 is 5. Factorising 21 and 35 are done similarly, following the appropriate rules for each value. For example for 21 the rule is "two by three is six plus two", so the 2 becomes a 6 and 2 is added to the 1 to get 63. The rule for three is "three by three, forward one", so we cancel the three and add 1 to the 6 to get 7, giving 21 ÷ 3 = 7[4].

Factorising the RSA-2048 value with an abacus presents more of a challenge since it contains 616 digits while a common abacus size is 9, 11, or 13 columns (digits) [30]. In fact the size of a 616-digit abacus is unpleasant to contemplate, so we leave the construction of such a bignum abacus to the reader, or perhaps an enthusiastic woodworking hobbyist with a YouTube channel.

6 Performing Quantum Factorisation operations with a Dog

As has been previously pointed out, the 2001 and 2012 quantum factorisation records may be easily matched with a dog trained to bark three times [31]. We verified this by taking a recently-calibrated reference dog, Scribble, depicted in Figure 6, and having him bark three times, thus simultaneously factorising both 15 and 21. This process wasn't as simple as it first appeared because Scribble is very well behaved and almost never barks. Having him perform the quantum factorisation required having his owner play with him with a ball in order to encourage him to bark. It was a special performance just for this publication, because he understands the importance of evidence-based science.

[4] If you're finding this difficult to follow without an abacus and a corresponding textbook, there are plenty of YouTube videos that show abacus division in practice.

Figure 6: Scribble preparing to replicate a quantum factorisation

The process was then repeated to have him bark five times, factorising the number 35 and thereby exceeding the capabilities of the quantum factorisation physics experiments mentioned earlier.

Unfortunately this process fails for the RSA-2048 values since the size of the factors exceeds even the most enthusiastic dog's barking ability. However there is another process that allows us to factorise even these huge numbers with a dog. Recall from section 4 that the prime factors p and q were either 2 or 6 apart. This led to an analysis where it was discovered that p = x – d and q = x + d, where x is the integer in the middle between p and q and d is either 1 or 3. It can thus be argued that d is the *real* secret. So teaching a dog to bark three times already gives us all of the actual factorisations with Shor's algorithm, plus 50% of the moduli in the D-Wave paper, in the same way that factorising 143 also factorises 56,153, 4,088,459, and 383,123,885,216,472,214,589,586,724,601,136,274,484,797,633,168,671,371.

We therefore propose the following process for factorising the RSA-2048 values using a dog. Initially, run the integer square root on N + 9. If this is a perfect square, have the dog bark three times. If the remainder is 8, have the dog bark once[5].

With this process we can now factorise all of the RSA-2048 moduli in the paper. We argue that the amount of sleight-of-hand used in this method is no more than typical quantum factorisation sleight-of-hand[6].

For the non-RSA-2048 values, given that two-digit composite numbers have at least one factor less than 10, we also estimate that factorising at least two-digit numbers should be within most dogs' capabilities, assuming the neighbours don't start complaining first. As did the VIC-20 and the abacus before it, this demonstrates that canine-based factorisation technology outperforms current physics-experiment based factorisation technology.

7 Proposed Quantum Factorisation Evaluation Criteria

Since all demonstrations of quantum factorisation to date have involved either sleight-of-hand numbers, sleight-of-hand preprocessing on a computer, or both, we propose the following standard evaluation criteria for future claims of quantum factorisation. These have the following properties:

- The factors are of a nontrivial size, 64 or 128 bits. This makes the problem space large enough that the solution can't be found through simple search techniques unrelated to factorisation.
- The factors are two prime values with a large difference between them and containing a 50:50 mix of 0 and 1 bits, randomly distributed. This prevents the construction of sleight-of-hand numbers that are readily amenable to factorisation via physics experiments.
- No preprocessing of the value to be factorised using a computer is permitted. This prevents the problem from being transformed into a different, easily-solved sleight-of-hand problem before the physics experiment even begins.
- The factors are unknown to the experimenters. This prevents the current practice of short-circuiting the factorisation process by taking advantage of knowing the answer before the process has even begun.
- The factorisation is performed on ten different values with the properties given above. This demonstrates repeatability of the process.

[5] This may require a dog other than Scribble, who has shown a tendency to report d = 3 regardless of the actual value of d. However since the sleight-of-hand factorisation using the Callas Normal Form covered in Section 3 always has p = 3, Scribble will have a success rate of 100% in this case.

[6] According to most Codes of Ethics, Scribble's contribution to this paper does not rise to the level where he is required to be listed as a co-author. However since he was a participant in the work rather than the subject of an experiment his contributions are exempt from review board approval.

As an aside, the above criteria also move the problem out of the space in which it is readily solvable using a VIC-20.

A process that meets these requirements is as follows:

1. The numbers to be factorised are generated by having a third party uninvolved with the factorisation take two randomly-generated 64- or 128-bit prime numbers with the properties given above and multiplying them together to obtain a 128- or 256-bit value.
2. The resulting value is then provided to the experimenters, with the two factors being discarded.
3. No computer preprocessing to convert the problem into an entirely different one is permitted.
4. A quantum factorisation is regarded as successful if it recovers the two factors for ten different arbitrary test values.

It should be noted that 128- and 256-bit values are completely insecure and should never be used in the wild, with the factors being recoverable on even a very low-powered device like a Raspberry Pi without much effort. Their use here serves purely as an easy, low-hanging-fruit target for quantum factorisation physics experiments to replace the current universal use of sleight-of-hand numbers that fall easily even to canine-based factorisation technology.

We anticipate that, as with current sleight-of-hand factorisations, researchers will in the future construct more sophisticated sleight-of-hand manipulations to allow even these rules to be bypassed. We therefore expect that updates to these rules will need to be made in the future as they are penetration-tested by quantum factorisers.

8 Future Work

After our successful factorisation using a dog, we were delighted to learn that scientists have now discovered evidence of quantum entanglement in other species of mammals such as sheep [32]. This would open up an entirely new research field of mammal-based quantum factorisation. We hypothesise that the production of fully entangled sheep is easy, given how hard it can be to disentangle their coats in the first place. The logistics of assembling the tens of thousands of sheep necessary to factorise RSA-2048 numbers is left as an open problem.

8.1 Replication Guide

In order to allow others to replicate our work, we have made all of the code used publicly available [26]. Although it was run on a cycle-accurate simulator in order to provide precise cycle counts for each factorisation, actual 6502 hardware with the capabilities of a VIC-20 is still being produced and could be used [33][34][35]. Since there is lots of ROM space left with some of the given hardware, one could even put WOZMON and Microsoft Basic on it.

Replication of the abacus-based portion of the work may be performed using any standard abacus. Since only two or three columns are required for the replication of the quantum factorisations of 15, 21, and 35, any abacus of size 9, 11, or 13 columns (digits) may be employed.

Finally, the apparatus for the canine-based factorisation may be obtained from any animal shelter. Although our experiment used a Staffy, almost any dog breed should be suitable, although the smaller yappy dogs may over-report values.

9 Conclusion

In this paper we showed how to replicate current quantum factorisation records using first a VIC-20 8-bit home computer from 1981, then an abacus, and finally a dog. In terms of comparative demonstrated factorisation power, we rank a VIC-20 above an abacus, an abacus above a dog, and a dog above a quantum factorisation physics experiment. Finally, we provided standard evaluation criteria for future claimed quantum factorisations.

10 Acknowledgements

The authors thank Peter Heinrich of the Zurich University of Applied Sciences for valuable information on the application of the Callas Normal Form in quantum factorisation, and Jon Callas and members of the Metzdowd cryptography list for their input and feedback.

11 References

[1] "Discrete logarithms and factoring", Peter Shor, *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, 1994, p.124.

[2] "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance", Lieven Vandersypen, Matthias Steffen, Gregory Breyta, Costantino Yannoni, Mark Sherwood and Isaac Chuang, *Nature*, **Vol.414**, p.883 (2001).

[3] "Experimental realization of Shor's quantum factoring algorithm using qubit recycling", Enrique Martín-López, Anthony Laing, Thomas Lawson, Roberto Alvarez, Xiao-Qi Zhou and Jeremy O'Brien, *Nature Photonics*, **Vol.6**, p.773 (2012).

[4] "Experimental study of Shor's factoring algorithm using the IBM Q Experience", Mirko Amico, Zain Saleem and Muir Kumph, *Physical Review A*, **Vol.100** (8 July 2019), `https://journals.aps.org/pra/abstract/10.1103/PhysRevA.100.012305`.

[5] "New largest number factored on a quantum device is 56,153", Lisa Zyga, 28 November 2014, `https://phys.org/news/2014-11-largest-factored-quantum-device.html`.

[6] "The Mathematical Trick That Helped Smash The Record For The Largest Number Ever Factorised By A Quantum Computer", The Physics arXiv Blog, 2 December 2014, `https://medium.com/the-physics-arxiv-blog/the-mathematical-trick-that-helped-smash-the-record-for-the-largest-number-ever-factorised-by-a-77fde88499`.

[7] “Chinese Researchers Break RSA Encryption with Quantum Computer: A Wake-Up Call for Cybersecurity”, Kalab Tenadeg, Medium, 19 October 2024, https://medium.com/@kalabtenadeg/chinese-researchers-break-rsa-encryption-with-quantum-computer-a-wake-up-call-for-cybersecurity-813247dd9585.

[8] “A First Successful Factorization of RSA-2048 Integer by D-Wave Quantum Computer”, Chao Wang, Jingjing Yu, Zhi Pei, Qidi Wang and Chunlei Hong, *Tsinghua Science and Technology*, **Vol.30**, **No.3** (June 2025), p.1270.

[9] “Digital Signature Standard (DSS)”, FIPS 186-5, National Institute of Standards and Technology, February 2023.

[10] “Factoring the largest number ever with a quantum computer”, Craig Gidney, 1 April 2020 (the date refers to the “factorisation” of a six thousand digit number in the latter part of the writeup, not the analysis portion), https://algassert.com/post/2000.

[11] “Quantum computer sets new record for finding prime number factors”, Leah Crane, *New Scientist*, 13 December 2019, https://www.newscientist.com/article/-2227387-quantum-computer-sets-new-record-for-finding-prime-number-factors.

[12] “Analyzing the performance of variational quantum factoring on a superconducting quantum processor”, Amir Karamlou, William Simon, Amara Katabarwa, Travis Scholten, Borja Peropadre and Yudong Cao, *Quantum Information*, **Vol.7**, Article No.156 (2021).

[13] “Pretending to factor large numbers on a quantum computer”, John Smolin, Graeme Smith and Alex Vargo,29 January 2013, https://arxiv.org/pdf/1301.7007.

[14] “The algorithm of the new quantum factoring record 1,099,551,473,989”, François Grieu, 15 December 2019, https://quantumcomputing.stackexchange.com/-questions/9204/the-algorithm-of-the-new-quantum-factoring-record-1-099-551-473-989.

[15] “Largest integer factored by Shor’s algorithm?”, François Grieu, 5 January 2023, https://crypto.stackexchange.com/questions/59795/largest-integer-factored-by-shors-algorithm/59796.

[16] “Quantum Factorization of 143 on a Dipolar-Coupling NMR system”, Nanyang Xu, Jing Zhu, Dawei Lu, Xianyi Zhou, Xinhua Peng, Jiangfeng Du, *Physical Review Letters*, **Vol.108** (30 March 2012), https://journals.aps.org/prl/abstract/-10.1103/PhysRevLett.108.130501.

[17] “Quantum factorization of 56153 with only 4 qubits”, Nikesh Dattani and Nathaniel Bryans, 25 November 2014, https://arxiv.org/abs/1411.6758.

[18] “Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer”, Avinash Dash, Deepankar Sarmah, Bikash Behera, Prasanta Panigrahi, 26 May 2018, https://arxiv.org/abs/1805.10478.

[19] “Largest integer factored by Shor’s algorithm?”, François Grieu, June 2018, `https://crypto.stackexchange.com/questions/59795/largest-integer-factored-by-shors-algorithm/59796#59796`.

[20] “A fast quantum mechanical algorithm for database search”, Lov Grover, *Proceedings of the 28th Symposium on Theory of Computing (STOC’96)*, July 1996, p.212.

[21] “Re: [Cryptography] Has quantum cryptanalysis actually achieved anything?”, Jon Callas, posting to the cryptography@metzdowd.com mailing list, message-ID `BFC54170-DDC4-4292-9A75-377B6D85406B@callas.org`, 19 February 2025.

[22] “Implementation of Shor Algorithm: Factoring a 4096-Bit Integer Under Specific Constraints”, Abel Chen, 7 April 2025, `https://arxiv.org/abs/2505.03743`.

[23] “Oversimplifying quantum factoring”, John Smolin, Graeme Smith and Alexander Vargo, *Nature*, **Vol.499**, p.163 (2013).

[24] **“**First Draft of a Report on the EDVAC**”**, John von Neumann, University of Pennsylvania technical report, 30 June 1945.

[25] **“**Hacker’s Delight**”**, Henry Warren, Jr., Addison-Wesley. 2002.

[26] “Code to factor the 2048-bit RSA "moduli"”, Stephan Neuhaus, `https://codeberg.org/sten13/rsa6502`.

[27] “The cc65 cross-compiler suite”, `https://github.com/cc65/cc65`.

[28] “Fake6502 CPU emulator core v1.1 “, Mike Chambers, 2011, `http://www.rubbermallet.org/fake6502.c`.

[29] “Square root by abacus algorithm”, Martin Guy, June 1985, `http://medialab.freaknet.org/martin/src/sqrt/sqrt.c`. This references a publication by a Mr.C.Woo who is listed as the co-author of a later edition of the abacus book referenced below.

[30] “Fundamental Operations in Bead Arithmetic”, Kwa Tak Ming, 1922, `https://archive.computerhistory.org/resources/access/text/2016/12/B1671.01-05-01-acc.pdf`.

[31] “Why quantum cryptanalysis is bollocks: A lesson from history”, Peter Gutmann, 2024, `https://www.cs.auckland.ac.nz/~pgut001/pubs/bollocks.pdf`.

[32] “CERN scientists find evidence of quantum entanglement in sheep”, Naomi Dinmore, 1 April 2025, `https://home.cern/news/news/cern/cern-scientists-find-evidence-quantum-entanglement-sheep`.

[33] “Build a 6502 computer”, Ben Eater, `https://eater.net/6502`.

[34] “W65C134SXB - 6502 based Microcomputer Board”, Western Design Centre, `https://wdc6502store.com/products/w65c134sxb-single-board-computer`.

[35] “Pocket265”, Aleksander Kamiński ,
`https://github.com/agkaminski/Pocket265`.