Computer Science Extended Essay

To what extent do contemporary quantum computing solutions affect the current and future security of RSA as a public key cryptography solution?

Candidate Code: JKG-660

Word Count:3680

Contents

Introduction 3
Background Information 5
Traditional Computing 5
Superposition 5
Quantum Computing 5
Contemporary Cryptography 6
Trapdoor Functions 6
Semiprimes and Multiprimes 7
Semiprime Factorization 7
Symmetric Encryption 8
Asymmetric encryption 9
RSA 9
The RSA Assumption 10
Shor's Algorithm 11
Neven's Law 11
Methodology 12
Investigation 13
Procedure in Steps 14
Data Presentation 15
Analysis 16
Conclusion 19
Limitations 20
Bibliography 22
Appendix 24
Appendix 1 24
Appendix 2 25

Introduction

Cryptography is an ever-growing topic of importance within our increasingly digital world. As more and more of our lives become digital, we must think of how we secure our privacy and security of information from bad actors. The methods that we rely upon in cryptography to secure our information are instrumental in ensuring that integrity of privacy at both a state and personal level remain trusted.

The advent of quantum computing has been troubling for many cryptographers, due to the inability for many trapdoor functions to function as effectively as before. While quantum computing is currently in its infancy, Neven's law, seen as a quantum successor for Moore's law, stipulates that the number of qubits within quantum computers will grow at a doubly exponential rate[1].

Furthermore, this double exponential growth rate would lead to sudden access to cheap quantum computing power for public use. Given that this advancement would also allow for home quantum computing solutions, decryption algorithms could be run by home users, jeopardizing the safety of information sent with unsafe encryption algorithms. **"nobody knows exactly when quantum computing will become a reality, but when and if it does, it will signal the end of traditional cryptography"** [2].

In a world where everything from state secrets to financial information is transmitted, the importance of cryptography grows by the day. While information such as financial data has a relatively short relevance period, other types of information have a much longer relevance

[1] Harnett
[2] Davis

period, such as state secrets, which can remain relevant for decades. Information such as this must remain safeguarded even long after the advent of future technology.

While current cryptographic standards such as standard RSA may not be quantum resistant, other quantum resistant encryption algorithms exist, such as symmetric key quantum resistance, which utilizes sufficiently large key sizes to resist quantum computing based attacks[3].

[3] Perlner

Background Information

Traditional Computing

Traditional computing refers to the usage of Boolean logic using transistor-based logic gates to implement algorithmic solutions to any goal. Traditional computing is used in all commonly found computers, and apply most invented algorithms[4]. Traditional computing uses binary bits that can be in two possible states, 1 or 0[5].

Superposition

Superposition is the ability for particles to hold multiple states until measured. When measuring the state of a particle in superposition, the probability collapses into one state, according to the probabilities of both[6]. Two particles in identical superpositions may be measured to have different states. This is in contrast to traditional computing, in which a bit's state is fixed, unless directly changed.

Quantum Computing

Quantum computing is the use of quantum phenomena in order to perform computation. Quantum computers make use of qubits, or entangled particles that are in a superposition of both 1 and 0, probabilities determining their state after measurement. For example, Qubit 1 is initialized with an equal probability of 1 and 0. After an arbitrary operation, the probability results in a 20% probability of 1 and an 80% probability of 0. After the qubit is measured, the qubit collapses to one position.

[4] Computing
[5] Katwala
[6] Superposition

Quantum computing and traditional computing differ in that quantum computing can simultaneously solve many equations at once, with the probability of the correct result constructively interfering towards the correct result. After the algorithm is complete, ad the qubit is measured, the qubit collapses to the correct result[7].

Contemporary Cryptography

Contemporary digital cryptography is largely built around asymmetric keys and unkeyed cryptosystems. Contemporary cryptography can be split into three main branches, symmetric encryption, asymmetric encryption and hashing[8]. It is largely built upon the concept of trapdoor functions, such as multiplication of primes. These allow for the two intended parties to communicate and secure their messages with relatively low computational requirements, while the adversary must employ significant computational power to decrypt the messages.

Trapdoor Functions

Trapdoor functions are mathematical functions that are easily calculated one way, however are computationally expensive to reverse[9]. The simplest example of a trapdoor function is factorization[10]; the multiplication of two primes such as $13 \times 17 = 221$ is easy even for a human, however the factorization of 221 without prior knowledge of one of the factors requires trial and error, which increases exponentially with the increase of the prime factors used.

[7] Katwala
[8] Oppliger
[9] Oppliger
[10] Oppliger

Semiprimes and Multiprimes

Semiprimes are the products of two primes. The generalization of a semiprime is defined as the product of two primes. For $n = pq$, with p and q distinct, the congruence: $p^q \equiv p(mod\ n)$ is satisfied[11]. For example,

$$p = 3$$

$$q = 5$$

$$n = pq = 15$$

Multiprimes are the products of multiple primes. While all non prime numbers are technically multiprimes, for the purposes of this essay, multiprimes will be defined as an extension of semiprimes. For example,

$$p = 3$$

$$q = 5$$

$$b = 7$$

$$n = pqb = 105$$

Semiprime Factorization

Semiprime factorization is the computationally expensive reverse function of prime multiplication. The multiplication of primes and its inverse function of factorization are trapdoor functions; the multiplication operation has a complexity of $\mathrm{O}(n \log n)$ using the

[11] Weisstein

Harvey-Hoeven algorithm[12], while the most efficient algorithm for the factorization of large semiprimes, knows as the general number field sieve (GNFS) has a complexity of $O\left\{\exp\left[c(\log n)^{\frac{1}{3}}(\log\log n)^{2/3}\right]\right\}$ [13]. Even this algorithm is slow on contemporary hardware for long public keys, such as the keys typically used in RSA encryption. Semiprime factorization is inherently complex because the possible factors of n include all primes between 3 and $\sqrt{n}$. As the private key length grows, the public key factorization time grows exponentially.

Symmetric Encryption

Symmetric encryption is the encryption of data using one key, this encryption method relies upon one key for both encryption and decryption, the nature of symmetric encryption requires both parties to be trusted to keep the key secret[14]. Given that the key may be intercepted, messages from both parties can be decrypted by an adversary. The key also must be distributed through a secure channel prior to any transmission of encrypted messages.

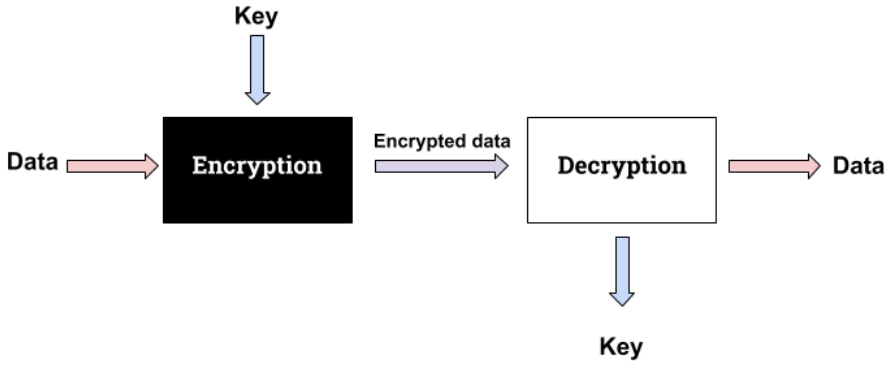

Figure 1: Visual representation of symmetric encryption

[12] Harvey
[13] Weisstein
[14] Oppliger

Asymmetric encryption

Asymmetric encryption is the usage of two keys to convert data to and from an encrypted state. The application of asymmetric encryption is that a public and a private key can be sent out. This public key is broadcast out and is used to encrypt the data being sent to the sender of the private key, which is the only key that can decrypt data encrypted with the public key[15]. The keys are inverses of the other, meaning that each key is the sole way to decrypt the encrypted data of the other. This asymmetry allows for communications to and from the sender to be encrypted, allowing for greater security than a symmetric key due to the lack of trust required between parties, and the ability for messages to remain private, without an initial trusted channel.

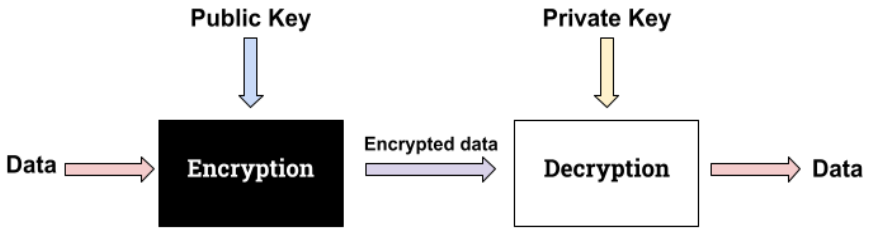

Figure 2: Visual representation of asymmetric encryption

RSA

RSA is an asymmetric cryptosystem in which user A publishes a public key which is the product of two large primes, also known as a large semiprime, along with a secondary value, known as the e. e is an integer that is coprime to the totient function of the public key. The private key is comprised of the factors of the public key, therefore, by factorizing the public key, the

[15] Oppliger

messages being sent to user A could be decrypted[16]. RSA is typically used to exchange a symmetric encryption key, which can encrypt and decrypt data much quicker than asymmetric encryption and decryption. The key to RSA's security lies in the difficulty of factorization of the public key. If an algorithm were discovered to easily factor semiprimes, RSA would be rendered insecure. Currently, RSA is considered secure if the key size, n is sufficiently large.

RSA has many variants, such as multiprime RSA(MPRSA), which utilizes multiple secret keys to increase complexity to adversarial attacks[17]. MPRSA is not currently widely used due to the establishment of other semiprime RSA standards.

The RSA Assumption

The RSA assumption refers to assumption that it is computationally infeasible to an adversary to factorize a sufficiently large RSA public key. The RSA assumption relies upon the factorization of the public key being a trapdoor function, however by cheaply computing the factors of the RSA public key, the private key of the encryption would be known, jeopardizing the security of the encryption. Given that adversary C can cheaply factorize user A's public key, user A has no method of distributing a symmetric key with user B, rendering all previously encrypted data by user A available to adversary C.

[16] Oppliger
[17] Kamardan

Shor's Algorithm

Shor's algorithm is a quantum algorithm for integer factorization that finds prime factors. Shor's algorithm is completed with both a quantum computer, and a traditional computer. The quantum part of the algorithm uses the inverse quantum Fourier transform to evaluate all states of the function simultaneously[18]. Due to contemporary encryption's reliance upon trapdoor function such as multiplication, their security lies in the time it would take for a traditional computer to factorize the results. Quantum computers bypass this security, by reverting the trapdoor function's super polynomial big O, to a polynomial time big O. Shor's algorithm runs with complexity[19]:

$$\mathrm{O}((\log n)^2(\log\log n)(\log\log\log n))$$

The current qubit requirement for Shor's algorithm factorization is $2n + 3$ qubits[20], with n as the number of bits required to represent the number. This threshold means that, with the current leading quantum computer, Jiuzhang has processor with 76 qubits[21] could factor a key of 36 bits.

Neven's Law

Neven's law is the quantum counterpart to Moore's law. While Moore's law predicts that the number of transistors on a given surface area of a chip will double every two years, an exponential growth, Neven's law predicts that the number of available qubits within a quantum

[18] Aaronson
[19] Baumer
[20] Beauregard
[21] Conover

computer will be doubly exponential, or that $\frac{dy}{dx} f(x) = b^x$. This suggests that the accessibility of quantum computers will reach a critical point, after which quantum computing will grow at such a rate that decryption solutions will become widely accessible to bad actors globally. While Neven's law does not directly affect the present, it does have implications for the future, suggesting that the accessibility of quantum computing will unexpectedly reach the mainstream, at speeds unrivaled by even traditional computing.

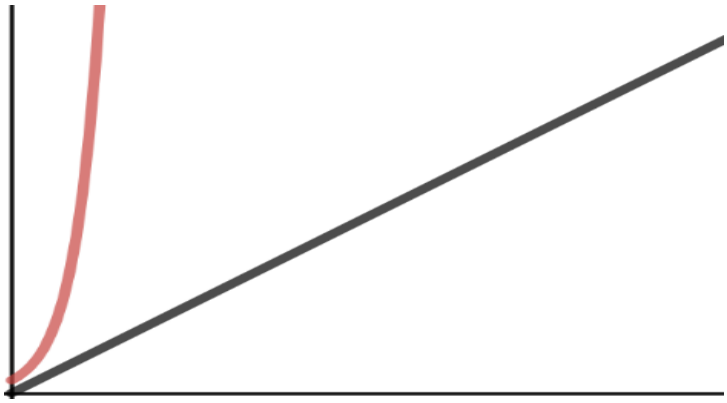

Figure 3: Double exponential growth(red) vs exponential growth(black) on a logarithmic y scale

Methodology

In carrying out this investigation it was imperative to first create a program to compare the results of the quantum computer to. The program was written in python, because the modules that interact with the IBM quantum experience (IBMQE) were also written in python, using the Qiskit module, avoiding possible conflicts. The results of this code would then be compared to the results of the Qiskit's built in Shor's algorithm function. For the traditional computing counterpart, the sympy module's factorint() function was used, because it already offered a high-performance solution to factorization of integers, reducing the time spent on coding the project.

The utilization of IBMQE provided a limitation of latency when communicating commands to the quantum computer. Furthermore, the IBMQE operates upon a fair-share basis[22]. Fair-share executes queued jobs on the system based upon system time allocations. Due to the limited access to the IBMQE system, the commands sent had no control over the priority queuing. In order to bypass this limitation, jobs were sent to the QUASM simulator, which simulates a quantum system. The QUASM creates multiple instances of the OpenQUASM simulator to concurrently simulate quantum circuits. While this may seem to invalidate the results of the experiment, the usage of a real quantum computer would introduce even more limitations to the experiment.

The queue length for ibmq_16_melbourne, the only quantum system with enough qubits to run Shor's algorithm on a number greater than 3 averages at about 6500[23]. Because IBMQE computers can only run one circuit at a time, this would mean lengthy queue times and an unreasonable time required to run each experiment. The systematic error of the queue length would completely invalidate any results produced by the tests. Furthermore, the lack of the ability to reserve system time due to a lack of affiliation to the IBM Quantum network meant that reserving system time for the experiment was an impossibility.

Investigation

Using the time package's perf_counter() function, the start time and end time of the two functions would be compared in order to extrapolate the time taken for both computers to factorize the number. The perf_counter() function returns the value of the clock with the

[22] IBM
[23] IBM

highest resolution available within the computer. The computer used for this experiment has a Ryzen 3600 CPU, running at 3.6GHz. The CPU used for this experiment is typical of contemporary enthusiast computers, so should provide an example of typical home decryption(factorization) solutions.

Procedure in Steps

1. The time to factor each odd integer from 3 to 31 was recorded using the Shor function of Qiskit five times[24].
2. The time to factor each odd integer from 3 to 31 was recorded five times using sympy.factorizeint() in python then moved into an excel spread sheet[25].
3. Record results to Excel spreadsheet.

[24] Appendix 2
[25] Appendix 1

Data Presentation

The data collected in each run was recorded in an excel spreadsheet and the average was taken of the five runs to reduce random error.

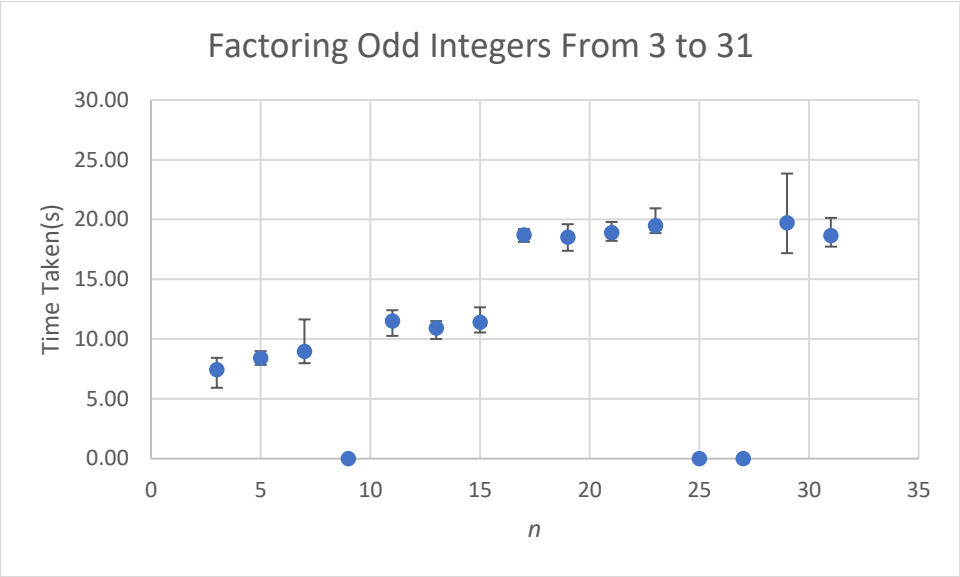

Figure 4: The initial data set of time taken to factorized odd integers between 3 and 31

The scatter plot of the data initially showed what appeared like one set of data, with random decreases of time taken to factor, however upon further inspection, the seemingly random dips in time taken to factor were correlated with the number of factors a number had. By restricting the data into two sets of, $\{2n-1 \mid n \in \mathbb{N}\} - \{(2n-1)^2 \mid n \in \mathbb{N}\}$ or set 1, and $\{(2n-1)^2 \mid n \in \mathbb{N}\}$ or set 2, then rerunning the factorization algorithms with their respective input restriction. While the quantum computing counterpart had large data variance, the traditional computing counterpart was consistent in its results, only varying slightly between runs, so the quantum data set was given error bars corresponding to the difference between the average runs, and the biggest and smallest runs. The two resultant datasets were as follows:

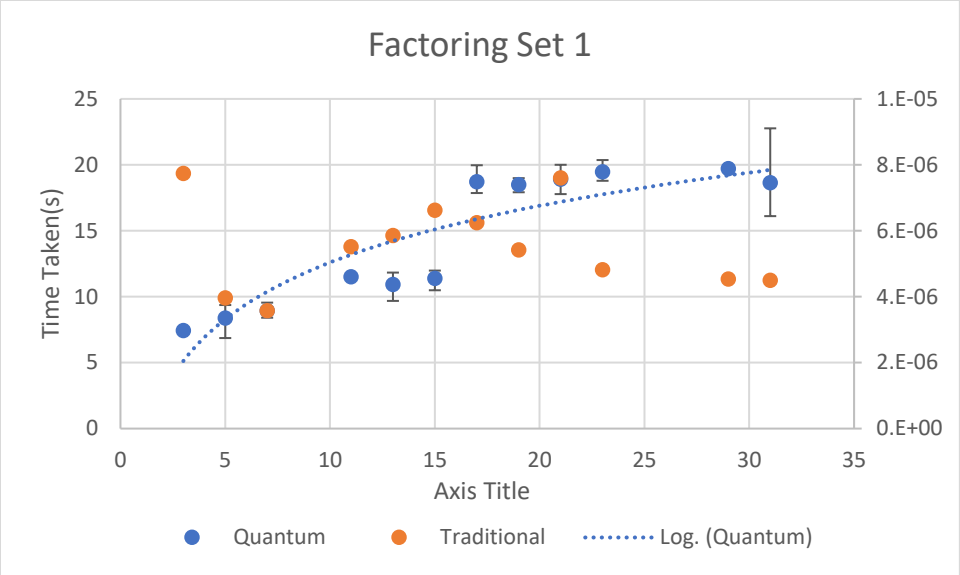

Figure 5: The factorization of set 1 was completed much faster by the traditional computer. The Quantum dataset uses the primary axis, while the traditional computing dataset uses the secondary axis.

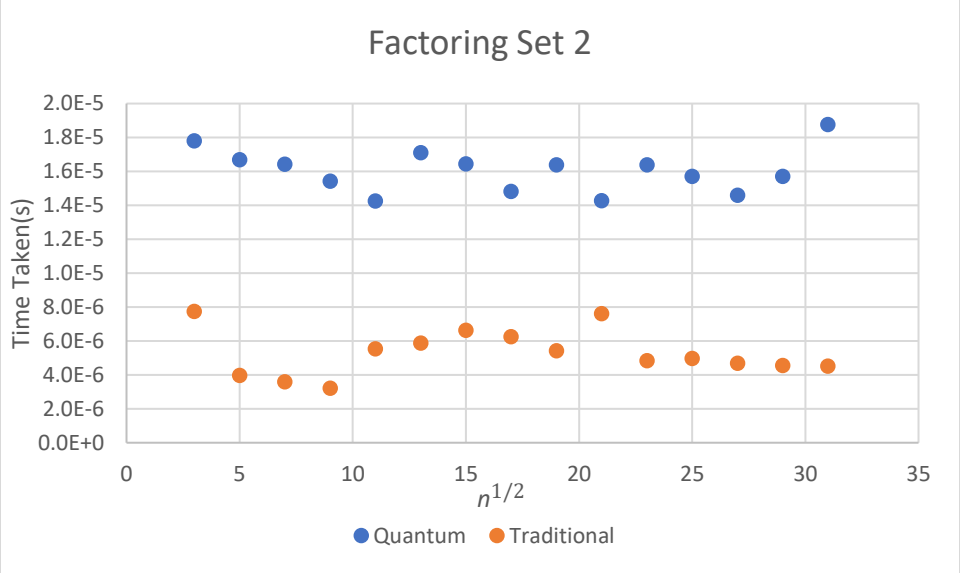

Figure 6: The factorization of set 2 was completed slightly faster by the traditional computer. Both data sets use the primary axis, completing the factorization in comparable times.

The change in speed of factorization of set 1 and 2 is ~ 733,301x. A speed up of 733,301 times suggests that the increase of factors of the public key would greatly increase the quantum resistance.

Analysis

The scatter plot of the data immediately showed that the factorization time of Shor's algorithm is dependent upon the number of prime factors of the input. While the usage of Shor's

algorithm in the factorization of n^2 was able to stay roughly similar in time to the usage of traditional computing, being slower by a factor of ~2. The factorization of primes and semi primes was much slower using Shor's algorithm compared to traditional computing, by a much greater factor of ~2,339,263. This suggests that contemporary quantum computing solutions are currently inferior in factoring compared to traditional computing solutions, suggesting that contemporary decryption and factoring have yet to be surpassed by their quantum counterparts. Quantum computing has not surpassed the supremacy threshold for factorization, meaning that as of now, quantum computing has not surpassed the threat level of contemporary decryption solutions, and can be considered safe.

After the initial tests with prime powers and semiprime numbers was completed, a test value of 105, the product of 3, 5 and 7 was factorized. The resultant data was divided by the size of the number being factorized, in order to reduce the systematic error incurred from increasing key size.

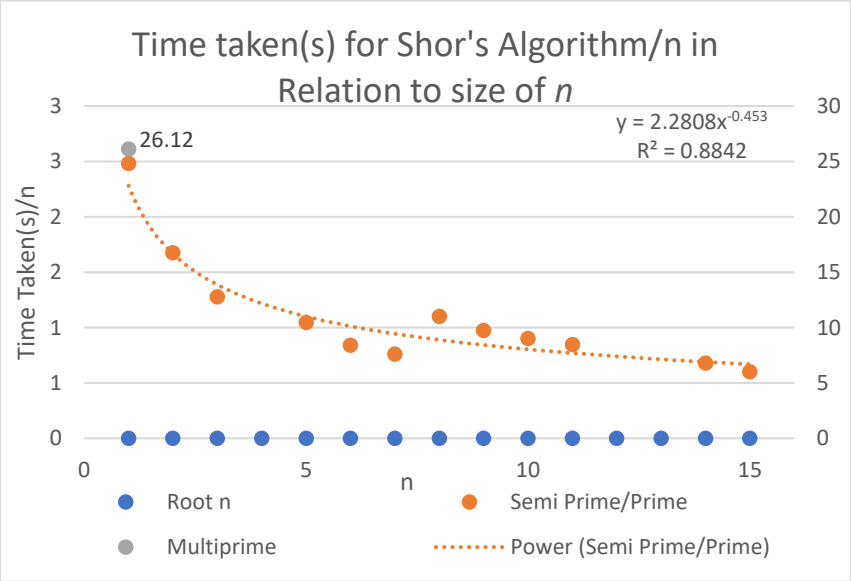

Figure 7: The perfect square data series and semiprime/prime data series utilize the primary axis while the multiprime data series utilizes the secondary axis

The inverse root decay of the curve suggests that the effectiveness of key length as a means of stopping Shor's algorithm approaches an asymptote. This in turn points towards the usage of key length as a means to dissuade quantum factorization as a limited solution. Each subsequent addition to key length would lead to diminishing returns upon the effectiveness of the semiprimes as a trapdoor function. Due to the limitations of this experiment, a trend line for multiprimes was unable to be established.

While quantum computing does not pose a threat to contemporary encryption solutions, the current state cannot be considered as static in the rapidly advancing field. When Neven's law is taken into account, the computational complexity of quantum computing begins to pose a threat to the security and trust able to be placed into current encryption standards such as RSA.

Taking Neven's law into account, increase in factorization time with the number of prime factors suggests that multiprime (numbers that are the product of three or more primes) would be more suitable as RSA keys to resist quantum attacks than semi primes. In order to create a quantum resistant encryption scheme, the usage of multiprime keys with similarly sized large prime factors would increase the decryption time for quantum computers attempting to decrypt the key.

The usage of multiprime keys does not fundamentally change the RSA system. MPRSA has already been implemented, with some attacks already discovered. MPRSA is less vulnerable to current RSA attacks, however attacks that exploit the modulus factors into three or four primes

may exist[26]. The possible existence of undiscovered computationally cheap MPRSA specific attacks reduces the efficacy of MPRSA as a cryptographic solution.

Conclusion

While quantum computing solutions are currently slow in comparison to even home computers, what must be considered is their potential. With the world's growing pace of innovation, the future must be considered when considering the solutions of the present. While the threshold point of quantum computing is extremely hard to predict, there will likely come a year in which quantum computing gains an immediate and apparent relevancy in the lives of consumers, in a pace never before seen. The marketing possibilities of "quantum" are no doubt exciting for the litany of computing businesses that produce products for consumers. This bodes incredibly dangerous for information previously stated as having a long shelf life, due to the possibility for state secrets intercepted currently to be later decrypted, possibly inciting future tensions after decryption.

While previously sent data with quantum insecure methods cannot be resecured, future encryption should utilize encryption schemes that utilize multiprime integer keys, rather than semiprime keys, due to the scaling big O that quantum computer face as the number of factors within the multiprime increase. This however, is counterbalanced by the "weakest prime" problem, in which a public key is only as strong as its weakest prime. By increasing the size of the primes, the set of possible keys is reduced to a prime that is between 3 and $\sqrt[3]{n}$. The weakest of the primes would be most easily solved through GNFS. With a semiprime public key,

[26] Hinek

the greatest of the two primes that would be used in the hypothetical multiprime public key, increasing its resistance to decryption attacks by traditional computers.

The trade off of current and future security is relevant to the cryptographic application. While MPRSA may provide greater resistance to currently discovered RSA attacks, the possible existence of attacks that exploit the modulus' factorization into more than two primes may prove that MPRSA encrypted keys may be broken in the future[27].

Limitations

The usage of Neven's Law to predict the future of quantum computing is contested. Neven's Law is based upon observation over a short time frame[28]. Furthermore, Andrew Childs, the codirector of the Joint Center for Quantum Information and Computer Science at the University of Maryland has stated that "When looking at all the moving parts, including improvements on the classical and quantum sides, it's hard for me to say it's doubly exponential."

The limitations of using a Quantum simulator were previously discussed, however its usage was needed for this experiment, due to the greater limitations that would occur by using a real quantum computer through the IBMQE.

The factorization of small semiprimes with traditional computers is inconsistent with the real-world implementations of RSA. RSA keys are typically 512 to 4096 bits long. The keys used in the experiment were at maximum 5 bits long. The usage of small semiprimes was due to the limitations of the quantum aspect of the experiment. The exponential increase in big O of factorization using traditional computing was not seen due to the small semiprimes factorized.

[27] Hinek
[28] Harnett

In order to make a more substantial comparison, a real quantum computer would be needed, in addition to a greater number of qubits available to factorize larger primes.

The largest limitation of this the experiment stemmed from the current state of quantum computing technologies. A repetition of the experiment done in the future would provide more substantiated results, after the factorization of sufficiently large numbers could be completed. Currently, publicly available quantum computing solutions are insufficient in qubit size to factor even the most insecure RSA keys, leading to a large discrepancy between real world application and the results of the experiment. If the experiment was repeated in the near future, after a quantum computer with 1027 bits[29] becomes available to the public, numbers directly applicable to current 512-RSA could be determined, providing a more substantiated conclusion, however even less qubits would be required for more applicable results. Even the factorization of a 32-bit number would provide greatly more applicable data. Should this experiment be repeated, the factorization of greater multiprimes than 105 would provide data to create trendlines for multiprime RSA keys. The distinction of the three or more datasets would allow for greater analysis upon the efficacy of multiprime keys as a means to create quantum resistant asymmetric encryption.

[29] Beauregard

Bibliography

Baumer, Elisa, et al. *Quantum Device Lab*, 15 May 2015, qudev.phys.ethz.ch/static/content/QSIT15/Shors%20Algorithm.pdf.

Beauregard, Stephane. “Circuit for Shor's Algorithm Using 2n+3 Qubits.” *ArXiv.org*, 21 Feb. 2003, arxiv.org/abs/quant-ph/0205095.

Conover, Emily. “The New Light-Based Quantum Computer Jiuzhang Has Achieved Quantum Supremacy.” *Science News*, 3 Dec. 2020, www.sciencenews.org/article/new-light-based-quantum-computer-jiuzhang-supremacy.

Davis, Clifton, and Christoph F. Eick. “Emperor: Cheap Legal Secure Cryptography for the Web.” *Emperor*, 1 Feb. 1999, dl.acm.org/doi/10.1145/298151.298489.

“Fair-Share Queueing.” *IBM Quantum Experience*, quantum-computing.ibm.com/docs/manage/systems/queue/.

Harvey, David, and Joris Van Der Hoeven. *HAL*, 28 Nov. 2020, hal.archives-ouvertes.fr/hal-02070778/document.

Hinek, Jason M, et al. Springer-Verlag Berlin Heidelberg, 2003, *On Some Attacks on Multi-Prime RSA*.

Katwala, Amit. “Quantum Computing and Quantum Supremacy, Explained.” *WIRED UK*, WIRED UK, 27 Jan. 2021, www.wired.co.uk/article/quantum-computing-explained.

“Number Field Sieve.” *From Wolfram MathWorld*, mathworld.wolfram.com/NumberFieldSieve.html.

Oppliger, Rolf. *Contemporary Cryptography*. Artech House, 2005.

Perlner, Ray A., and David A. Cooper. “Quantum Resistant Public Key Cryptography: A Survey.” *NIST*, 10 Nov. 2018, www.nist.gov/publications/quantum-resistant-public-key-cryptography-survey?pub_id=901595.

“Quantum Superposition.” *Joint Quantum Institute*, jqi.umd.edu/glossary/quantum-superposition.

Shtetl-Optimized, www.scottaaronson.com/blog/?p=208.

Techopedia. “What Is Computing? - Definition from Techopedia.” *Techopedia.com*, Techopedia, 19 June 2012, www.techopedia.com/definition/6597/computing.

Appendix

Appendix 1

Shor's Algorithm

```
from qiskit import IBMQ
from qiskit.aqua import QuantumInstance
from qiskit.aqua.algorithms import Shor
import time
print("Imports Successful")

provider = IBMQ.load_account()

# Specifies the quantum device
backend = provider.get_backend('ibmq_qasm_simulator')
print("backend successful")

def factorizor(n):
    # Function to run Shor's algorithm where
    # n is the integer to be factored
    factors = Shor(n)
    starttime = time.perf_counter()

    result_dict = factors.run(QuantumInstance(
        backend, shots=1, skip_qobj_validation=False))

    endtime = time.perf_counter()
    timelapsed = endtime - starttime
    print(timelapsed)

factorizor(int(input()))
input()

```

Appendix 2

Traditional Factoring

```
import time, math
from sympy.ntheory import factorint
from xlwt import Workbook
print("imports completed")
# Workbook is created
wb = Workbook()

# add_sheet is used to create sheet.
newsheet = wb.add_sheet('Sheet1', cell_overwrite_ok=True)

i = 0
n = 3
while n < 32:
    newsheet.write(0, math.floor(n / 2), n)
    while i < 5:
        starttime = time.perf_counter()
        factorint(n)
        endtime = time.perf_counter()
        timelapsed = endtime - starttime
        newsheet.write(i, math.floor(n / 2), timelapsed)
        i += 1
    n += 2
    i = 0
wb.save('brim.xls')
print("done")
input()

```