Implications of Entropy on Symmetric Key Encryption Resilience to Quantum

TABLE
OF
CONTENTS

INTRODUCTION

Quantum computing leverages the quantum properties of entanglement and superposition to deliver a huge leap forward in computation to solve certain problems. Quantum computers operate using quantum entanglement called quantum bits or "qubits." These qubits can exist in a superposition of 0 and 1, allowing a quantum computer to compare multiple permutations simultaneously, thus making it possible to speed up the process of solving specific problems. Using quantum computers, certain computational problems can be solved within a short period of time that would ordinarily take a classical computer thousands of years to solve.

The security of cryptography relies primarily on the use of strong and secure keys. A key is considered strong if it is difficult to guess and truly random. Generally, information security standards require a minimum of 128 bits of security strength for cryptographic keys. This means that on average, an attacker would need to try at least half of the possible keys. In the worst-case scenario, they would need to try all possible keys, which would amount to trying a minimum of 2^{128} keys as the most efficient method of attack.

Future Quantum computers (QC) will use Grover's algorithm, also known as the quantum search algorithm. This quantum algorithm can efficiently search an unstructured database to find the unique input that satisfies a given criterion with high probability (i.e., a cryptographic key). This approach would weaken symmetric key cryptography, where a QC could determine the private key that was used to encrypt the data in a short period of time.

Current thinking assumes that doubling the key size from 128 bits to 256 bits will be sufficient to compensate for the increased efficiency of a quantum computer to make symmetric key encryption quantum resistant for at least the next 15 years. Unfortunately, unless the source of randomness "entropy" used to generate a private key is truly random and able to generate a random key length of 256 bits, then the effect by increasing the key length will be no more resistant to potential quantum attack.

What is Entropy and the Source of Randomness?

Encryption is a method of encoding data to protect it from unauthorized access. It uses a secret key to scramble the data in a way that makes it difficult for anyone to read the data without the key. One of the fundamental components of effective encryption is entropy, which is a measure of a key's randomness. If the key is not sufficiently random, it may be possible for an attacker to guess the key, which would compromise the security of the encrypted data.

The measure of entropy is typically in units of bits, and for entropy rate, bits per sample. In the context of cryptography, entropy is important for ensuring the security of data. The higher the entropy, the more random the data used to generate a key. This means that a lower entropy would make it easier for a quantum computer to predict the key used. Therefore, it is important to use sources of entropy that are highly random to ensure the security of cryptographic keys.

Cryptographic applications often rely on random data to function properly. This random data can be obtained from different sources, including physical random number generators (RNGs), software-based pseudorandom number generators (PRNGs), or a combination of both. The quality of the randomness, or entropy, of the data generated from these sources can vary. In recent years, the use of quantum random number generators (QRNGs) has become more popular because these devices can produce truly random data.

QRNGs use the unpredictable nature of quantum measurements to generate random strings of data. These devices typically consist of two main components: a quantum state generator and a measurement device. The generator produces a quantum state, which is then measured by the measurement device to produce a raw random string. This raw string may be further processed using privacy amplification techniques to "smooth out" any potential biases and produce a smaller, truly random, and uniform string that is independent of an adversary.

Therefore, even if the key length is increased to 256 bits, the randomness used to generate the cryptographic keys may not be sufficiently random. As a result, the encrypted data may be vulnerable to quantum attack sooner than expected.

When Doubling AES Key Size to 256 Has No Impact on Quantum Resistance

This paper quantifies the effect of poor entropy on AES 256-bit encryption. It shows that the run time of Grover's algorithm on a quantum computer to compromise the key is significantly reduced unless the source of entropy is in fact truly random.

Furthermore, as quantum computer gate fidelity improves over time, the number of logical qubits requires to run Grover's algorithm is further reduced. This combined effect with poor entropy could make symmetric key encryption less resistant to quantum attack in the near term. Serious consideration should be given to the source of entropy on a device before increasing the key length for AES, assuming that this would make it quantum secure.

QUANTIFY ENTROPY

In simple terms, entropy is a measure of uncertainty or randomness. In the context of quantum computing, von Neumann entropy [1] and quantum min-entropy [2] are commonly used to measure the uncertainty of a quantum state, which can be more complex than a classical state consisting of combinations of 0's and 1's. This is because qubits can exist in any linear combination of the states 0 and 1 (denoted as $|0\rangle$ and $|1\rangle$).

By measuring the von Neumann entropy or quantum min-entropy of a quantum state, we can quantify the amount of uncertainty or randomness and determine how many independent and uniform random bits can be extracted from the state. As an example, consider a biased coin that produces heads two-thirds of the time and tails one-third of the time. In this case, you would expect a result of heads roughly 67% of the time. The entropy would be lower than for a fair coin because the outcome is less uncertain or random. This has important implications for the security of random number generators used in cryptography because a biased generator could potentially be exploited by an adversary.

Von Neumann entropy and quantum min-entropy are two different but important entropy measures for quantum systems. Both relate to how many truly random bits (e.g., bits useful for a cryptographic secret key) one may extract from a random system. Von Neumann entropy gives an upper-bound to this, which is useful when dealing with theoretical systems, while min-entropy gives a practical bound. As its name implies, min-entropy is always no more than von Neumann entropy.

Extracting Randomness: Consider the following scenario: Alice has access to some randomness source (e.g., measuring a quantum state). However, this source is not perfect and may be biased, or an adversary may have partial control of the source. Let A be the random variable modelling Alice's source and E the adversary system Eve (which may be trivial if there is no adversary). In general, Alice may run her source through a privacy amplification process to "smooth out" the randomness in her string outputting a uniform random string S. In general, this process involves choosing a random two-universal hash function f, which takes as input an N-bit string and outputs an ℓ-bit string with $\ell \leq N$; then $S = f(A)$. Furthermore, it can be shown that Eve's information on the output string S may be made negligible.

The question, then, is what should ℓ be? If $\ell = N$ then the entire string A holds a uniform random string that is independent of an adversary (and thus may be used as a secret key for instance). The more information an adversary has about A, however, the smaller ℓ must be. As it turns out, ℓ is a direct function of the entropy of A.

Let $H(A|E)$ denote the von Neumann entropy and $H_\infty(A|E)$ the min-entropy of this source. If each bit of A is independent and identically distributed (i.e., $A = (A_0)^N$) for some random variable A_0, acting on one bit (that is A is N-independent copies of some random variable A_0) then it can be shown that [3]:

$$\lim_{|A|\to\infty} \frac{\ell}{|A|} = H(A_0|E) \tag{1}$$

That is, the von Neumann entropy measures how many uniform random bits one can extract from an Independent and Identically Distributed (i.i.d.) source in the asymptotic case (as the size of the bit string $|A|$ goes to infinity). On the other hand, one usually cares more about realistic finite signal cases where $|A| < \infty$. In this case, one must use min-entropy:

$$\ell = H_\infty(A|E) - 2\log\frac{1}{\epsilon}, \tag{2}$$

where ϵ is a security parameter that determines how far the final string is from uniform and independent.

For an example, let's consider sources that are independent of an adversary (that is, there is no adversary). The A random variable is always classical in these cases, so the von Neumann entropy actually agrees with the usual Shannon entropy. Let us also, for this example, consider an i.i.d. source, so $A = A_0^N$ where A_0 is a random variable that takes the value 0 with probability p and takes 1 with probability $1 - p$. In this case, we have:

$$H(A|E) = H(A) = -p\log p - (1-p)\log(1-p), \tag{3}$$

where all logarithms in this report are base two unless otherwise specified. The min-entropy, on the other hand, is:

$$H_\infty(A|E) = H_\infty(A) = -\log\max(p, 1-p). \tag{4}$$

Note that it always holds (for any random variable) that $H_\infty(A|E) \leq H(A|E)$. Thus, it is not sufficient in finite key scenarios to bound Shannon or von Neumann entropy. Instead, one must look at min-entropy as it is potentially smaller and produces a "worst-case" bound on the secret random string size).

For non-i.i.d. scenarios, where the A system takes the value a $\in \{0,1\}^N$ with probability p_a, the definition of Shannon entropy is:

$$H(A) = -\sum_{a\in\{0,1\}^N} p_a \log_2 p_a. \quad (5)$$

The definition of min-entropy is:

$$H_\infty(A) = -\log \max_a p_a. \quad (6)$$

Computing the above for arbitrary systems can be difficult and is usually the main point in a security proof of a cryptographic protocol of this nature. We do not go into those details here. However, the key takeaway is that entropy relates directly to how large a secret key one can extract from a source. The higher the entropy, the longer the key. If one is using a low-entropy source, the actual size of the key will be smaller than the number of bits in the source. One can then "stretch" the key using, for instance, a pseudo-random number generator. However, this does not increase the actual entropy in the final key.

2.1 Quantum Algorithms

Quantum algorithms work with qubits. As stated earlier, classical bits can be in a state of 0 or 1, whereas qubits can be in a superposition state of both 0 and 1. In more detail, a qubit can be in a zero state (denoted $|0\rangle$) or a one state (denoted $|1\rangle$) but also a superposition of both denoted $\alpha|0\rangle + \beta|1\rangle$. This can be extended to *n*-qubits: Given an *n*-qubit state, it may be in any classical *n*-bit state (such as $|01101\cdots\rangle$) or a superposition of all possible *n*-bit states, namely $\sum_x \alpha_x |x\rangle$. In the above superposition states, the values α_x are called probability amplitudes. Given a superposition state, one may measure it, which causes the state to collapse to a single *n*-bit string with probability $|\alpha_x|^2$.

A quantum algorithm is typically represented as a circuit, which is a collection of simple, basic operations that work on either a single qubit or across multiple qubits. A classical algorithm, of course, can be represented in the same way. However, a classical algorithm can work with only a single classical input at a time. (Parallel processes can of course increase this to some extent). A quantum algorithm can work with a single superposition state at a time . However, a superposition state can represent all possible *n*-bit classical input strings, so this means a quantum algorithm/circuit can "act on" all possible classical input states simultaneously. Of course, at the end of the day, one needs to perform a measurement.

Quantum algorithms generally operate in two stages:

> The first stage is to take your input data, prepare an initial quantum state based on this input data, and run a quantum circuit on it. A quantum circuit is a collection of elementary "gates" acting on the various qubits of the system. This circuit is, essentially, the step-by-step details of the quantum algorithm telling the quantum computer how to manipulate the underlying data.

> The second stage involves measuring the output of the first stage and performing some classical post-processing to interpret the result. Based on the output of this stage, one may have to run the algorithm again to get a conclusive result.

Both stages may be repeated multiple times before an answer to the problem can be found. The quantum circuit stage typically operates on a superposition state and manipulates the α_x values so that, the correct answer to the given problem is measured with higher probability than wrong answers.

2.2 Grovers Search

Grover's search algorithm is one of the fundamental algorithmic breakthroughs in quantum computing showing that a quantum computer can search through an unsorted database with a quadratic speedup. Let's assume we have a list of items X. In this list, there are N items, and we want to see if some item $x*$ is in the list. If the list is unsorted and there is no necessary structure to the list, a classical search algorithm would have to search through the entire list to see if $x*$ is in the list, then this algorithm would require about N operations to search the list to find $x*$. Whereas, using Grover's algorithm, a quantum computer can determine if $x*$ is in the list using only approximately $\sqrt{N}$ operations. To put this into context, if $N = 100$, a quantum computer would need only 10 steps to find $x*$, whereas a classical computer would require 100 queries on the list in a worst-case scenario.

An alternative, and more useful, formulation of Grover's algorithm is as follows: Let's assume we have a function f that takes as input an n-bit string and outputs 0 or 1. However, this function has the property that there is only one input string $x*$ such that $f(x*) = 1$ and for any other input $f(x*) = 0$. One is able to evaluate the function on any input x, and the goal is to find $x*$. This is exactly the problem we have when breaking AES in the Chosen-Plaintext Attack (CPA) model. We have a known message and ciphertext mapping (namely, we have a ciphertext c such that $c = AES(k*, m)$ for a known message m and unknown $k*$). We can define f to be a function that decrypts c given a candidate key k. The function outputs 1 if the decryption leads to the correct message (thus, only when $k = k*$).

Grover's search requires two quantum circuits: an oracle implementation of f and the Grover Diffusion Operator. The first is a quantum circuit U_f such that applied to the input x^* returns, in some fashion, a TRUE, while for any other input x will return FALSE. Due to the ability for a quantum circuit to act on a superposition of states, the algorithm will apply this single circuit to all possible input states simultaneously. Second, the diffusion operator causes a shift in the probability amplitudes to "favor" the correct state slightly. By this, we mean that the probability of measuring the wanted, yet currently unknown, $x*$ will increase slightly with each application of the diffusion operator. This is only a slight increase, however. Thus, the application of these two circuits must be repeated approximately $\sqrt{N}$ times before the probability of making the correct measurement increases substantially.

Figure 1 illustrates an example of Grover's algorithm for 3 qubits where the quantum circuit to solve the problem using a phase oracle is:

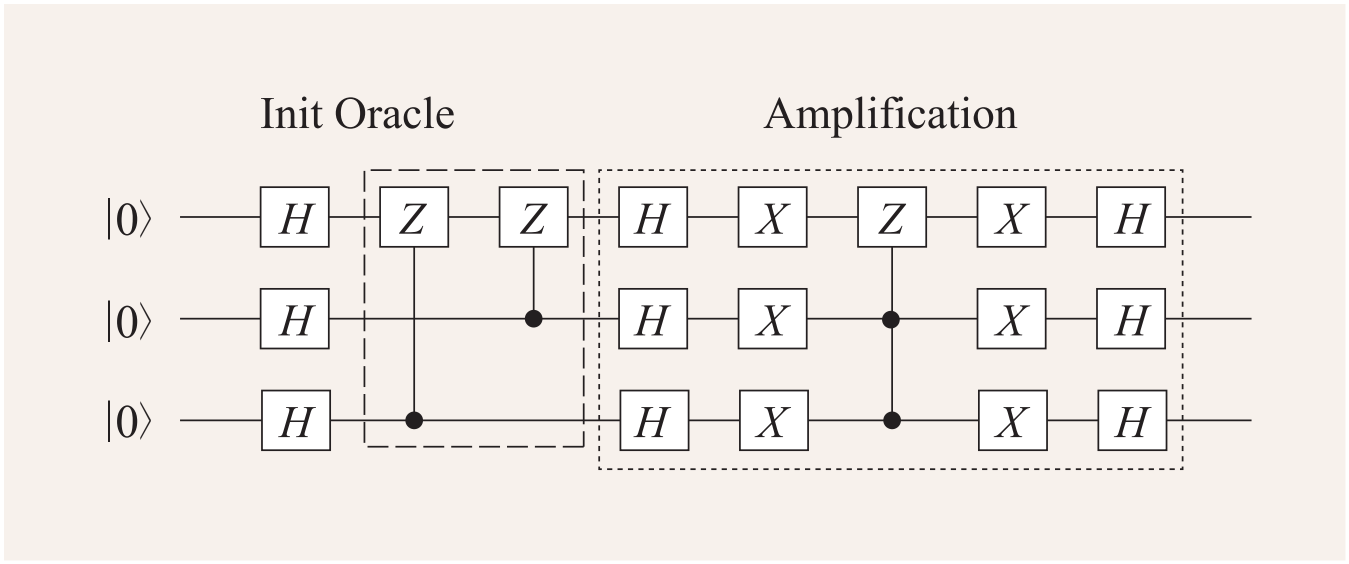

Figure 1: *Quantum Circuit for Grover's Algorithm*

Note: this is explained in more detail in IBM Qiskit textbook on Grover's Algorithm [4]

Let's assume that AES encryption is being utilized with a physical key size of n bits, where n may be 128, 192, or 256. Let's also assume an adversary holds a known ciphertext/plaintext pair that can be used by Grover's algorithm oracle to qualify the key result. Whereby, the adversary has (m, c) such that c = AES-Enc$(k*, m)$ (for an unknown secret key $k*$). This is not unrealistic and is, in fact, a potential within the CPA security model. Given this, we may construct the following function:

$$f(k) = \begin{cases} 1 & \text{if } Dec(k,c) = m \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

That is, $f(k)$ is a "check function." It will decrypt the given ciphertext using key k and see if it equals the known plaintext m. If it does match, then $f(k) = 1$ and the adversary knows that k is the correct key (i.e., $k = k*$). Otherwise, $f(k) = 0$ and k is not the correct key. Thus, a general brute-force-search attack against the AES encryption system would be defined as follows:

1. Set $k = 0 \cdots 0$.
2. Evaluate $f(k)$. If this is 1, then k is the correct key.
3. Otherwise, repeat Step 2 for $k = k + 1$.

Clearly the above attack's running time is exponential in n. For a 128-bit key, it will require approximately 2^{128} cycles to break the encryption scheme. However, based on our earlier discussion, we realize that this entire problem can be rephrased in terms of Grover Search. Indeed, $f(k)$ is our oracle evaluation.

Because $f(k)$ is basically running AES-Dec, and because any classical program can be translated to an efficient quantum circuit, we may construct a quantum version of this function. Namely, we may construct an efficient quantum circuit Of which takes as input n-qubits and performs the following computation:

$$Of\,|k\rangle = (-1)^{f(k)}|k\rangle \quad (8)$$

Namely, we have $Of\,|k\rangle = |k\rangle$ for all keys however $k \neq k*$; however $O_f\,|k*\rangle = -\,|k*\rangle$. This is the oracle function required for AES-Dec using Grover's algorithm.

3.1 Runtime Analysis

We now consider the running time of Grover's search for AES. Again, let n be the physical key-size used and we assume it is produced from a source with h-bits of entropy with $h \leq n$. Let $c128, c192, 2256$ be the running time of the AES decryption function when a physical key-size of 128, 192, or 256 bits is used. This directly determines how long Of will take to run also.

Now we need to compute the run time (n, h), namely how long the Grover's search algorithm is expected to run given a physical key size of n-bits produced from an h-bit entropy source. As discussed earlier, Grover consists of running a diffusion operation followed by the oracle M times. The first requires approximately $4n + n^2$ gates, while the second, as discussed, requires c_n time. We assume here that each gate takes 1 unit of time. Thus, the total time per round is $(4n + n^2 + c_n)$, and for M rounds, we require $M(4n + n^2 + c_n)$ time. Finally, n Hadamard operators are needed at the start, and n measurements at the end. Thus, the total run time for M rounds is $2n+M\,(4n+n^2+c_n)$. Now we can assume the worst case in that if $k*$ was drawn from a source with h its of entropy, then the total search space is only 2^h. Because M is approximately the square root of the search space, we conclude:

$$\text{Runtime } (n,h) = 2^{h/2}\,(n^2 + 4n + c_n) = 2n. \quad (9)$$

3.2 Required Number of Qubits

We now compute the estimated number of qubits needed to run the Grover's algorithm assuming imperfect gates. We assume surface coded qubits whereby one logical qubit (needed to run the algorithm) is actually encoded by approximately d^2 physical qubits. We assume each individual gate has a probability of error of 10^{-3}. Using d^2 physical qubits to represent a single logical qubit, the probability of an error, then, can be reduced to $10^{-\frac{d}{2}+1}$ per gate. From the above analysis, we have a bound on the total number of gates used. Thus, if one wants, at worst, a probability p of failure, one will require:

$$d = \log\left(\frac{2n + 2^{h/2}(n^2 + 4n + c_n)}{p}\right) + 2 \quad (10)$$

resulting in $(2d+1)^2$ physical qubits per logical qubit. Thus, the total number of qubits is:

$$\text{NumQubits}\,(\mathrm{n},\,\mathrm{n}) = n\left[2\log\left(\frac{2\mathrm{n} + 2^{h/2}(n^2 + 4n + \mathrm{c}_n)}{p}\right) + 5\right]^2 \tag{11}$$

3.3 Evaluation

Using the equations 9 and 11, we may evaluate the relative security of AES assuming keys are produced from low-entropy sources. Let's assume n = 256, meaning the physical key size is 256 bits. Where the key is produced from an entropy source h ranging from 32 bits to 256 bits. The expected running time of the algorithm is shown in Figure 2. Note the exponential nature of the runtime curve after the entropy exceeds that of approximately 220 bits. This shows how important it is not simply to have a physical key size of 256 bits, but also to ensure that it is produced from a high-entropy source. (Note that time = computational cycles.)

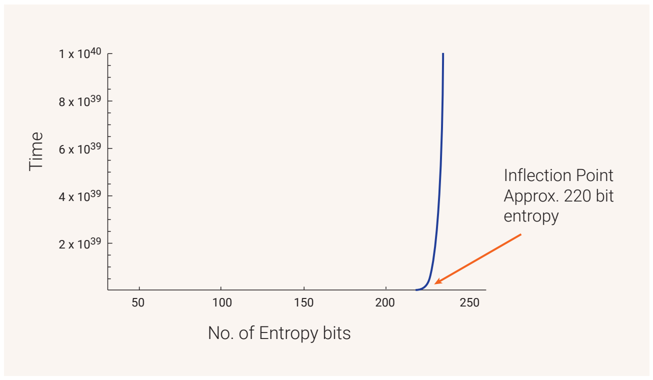

Figure 2: *Run Time of Grover's Algorithm to Break AES 256-bit Encryption Using a 256-Bit Key Across Entropy Range.*

The same plot is shown in Figure 3 at log scale. We also note that the physical key size matters far less than the entropy source.

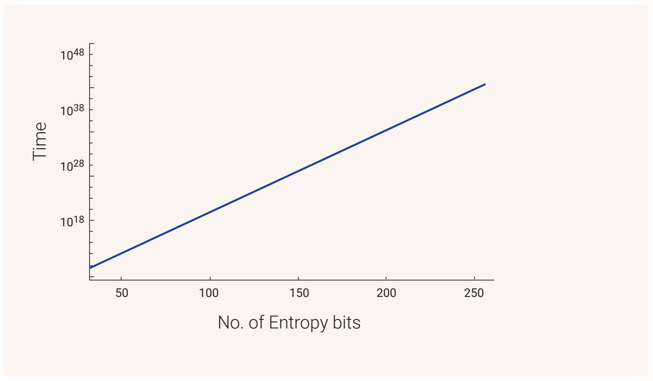

Figure 3: *Run Time of Grover's Algorithm to Break AES 256-Bit Encryption Using Different Key Sizes. (Log Scale)*

In particular, Figure 4 shows that increasing the physical key size from 128 to 256 leads to only a relatively small increase in Grover's algorithm attack run time as compared to increasing the entropy from 128 to 256. The blue line is a key of physical size 256 bits, while the yellow line is 128 bits. Notice that simply increasing the key size without increasing the entropy leads to only a small increase in running time as compared to Figure 2 which shows increasing the entropy leads to an exponential increase in running time. Thus, it is not sufficient to simply increase the physical key size without also increasing the entropy of the underlying key.

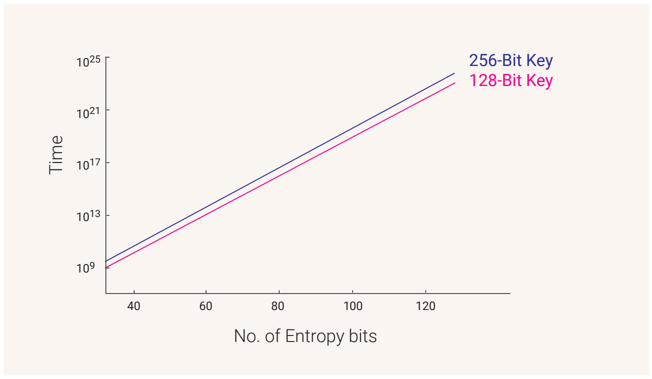

Figure 4: *Comparing the Running Time of Grover's Algorithm for a Key with a Certain Level of Entropy.*

The expected number of qubits needed to perform the attack is shown in Figure 5. Interestingly, decreasing the probability error rate per gate of Grover's attack does not lead to a significant decrease in the required number of qubits as compared to the increase needed when increasing the underlying entropy of the key.

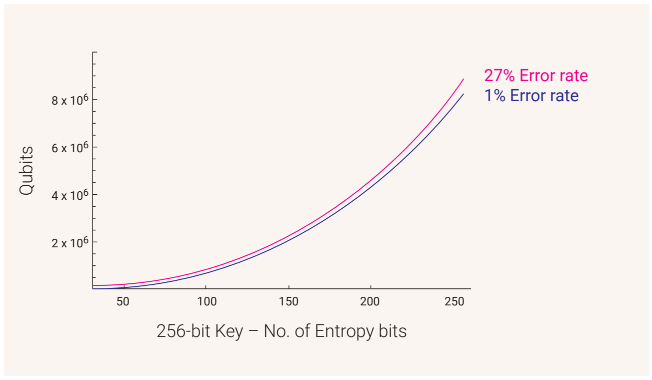

Figure 5: *Gate Fidelity – Impact of Required Number of Qubits to Perform Grover's Attack.*

CONCLUSION

The source and quality of entropy will have a significant impact on the security of symmetric key cryptography to be quantum resistant. This paper has illustrated that unless a source of entropy used for 256-bit symmetric key AES is truly random across at least 225 bits, it may have little effect in terms of being any more secure.

Although protocol standards are being changed to enable support for 256-bit symmetric key encryption, applying this across a plethora of devices already in operation may have some challenges. This may affect devices that:

> Are unable to be updated to support 256-bit symmetric key encryption and need to remain in operation using the current 128-bit or less encryption protocols.

> Can be updated to support 256-bit symmetric key encryption, but the source of entropy used to generate the key is poor, and even if updated will have no improvement in terms of being quantum secure.

> Can be retrofitted in the field with a true source of entropy (e.g., QRNG or entropy as a service). Additional CPU cycles are required to encrypt and decrypt messages with extended key lengths, so this would have implications for the application operating on the device (i.e., real-time, ultra-low-latency data).

Furthermore, as quantum computers continue to improve, leading to a reduction in noise, the fidelity of the gates used to perform algorithms will produce fewer errors. This will allow Grover's algorithm to be performed using a smaller number of qubits, making it more effective at compromising AES with low entropy. Yet as highlighted, the number of qubits required to break an AES-256 bit key is not significantly more than that required for a 128-bit key.

All of these considerations need to be made when assessing the risk of devices, applications, and services when planning transition strategies to be quantum resistant in the future. Effectively planning transition strategies to become quantum resistant in the future involves assessing their ability to adapt to new cryptographic techniques and standards as they emerge. This is necessary to protect against potential vulnerabilities that may be exposed by the development and advancement of quantum computing capabilities.

For more information about the ATIS Quantum-Safe Communications and Information initiative, visit https://www.atis.org/initiatives/quantum.

CONTRIBUTORS

Ian Deakin
Principal Technologist, ATIS

Walter Krawec
Assistant Professor of Computer Science and Engineering at the University of Connecticut

William Trost
Lead Member of Technical Staff, Quantum Computing Security Lead, CSO, AT&T

REFERENCES

[1] Von Neumman/quantum entropy: Nielsen, Michael A., and Isaac Chuang. “Quantum computation and quantum information.” (2002): 558-559.

[2] Quantum min-entropy: Renner, Renato. “Security of quantum key distribution.” International Journal of Quantum Information 6, no. 01 (2008): 1-127.

[3] Devetak, Igor, and Andreas Winter. “Distillation of secret key and entanglement from quantum states.” Proceedings of the Royal Society A: Mathematical, Physical and engineering sciences 461, no. 2053 (2005): 207-23

[4] IBM Qiskit textbook on Grover’s Algorithm https://qiskit.org/textbook/ch-algorithms/grover.html

COPYRIGHT AND DISCLAIMER

ATIS-I-0000093

Published February 2023