Deloitte.

Why Quantum Computers Aren't Cracking RSA Yet: A Practical Guide to Quantum Error Correction

Authors:

Itan Barmes PhD, Global Quantum Cyber Readiness Capability Lead at Deloitte Consultative Services B.V.

Colin Soutar PhD, Global Quantum Cyber Readiness Leader at Deloitte & Touche LLP

Alvaro Veliz Osorio PhD, Head of Tech Partnerships, Riverlane Ltd

Introduction

The rapid advancement of quantum computing is bringing businesses closer to real-world applications of this transformative technology. One significant application is solving the integer factorization problem using Shor's algorithm. Since integer factorization underpins the RSA (Rivest-Shamir-Adelman) and other cryptosystems, the rise of quantum computing could expose vulnerabilities in data security across sectors.

As quantum computing evolves, estimating when it might compromise current cryptographic protocols becomes crucial. These insights can help guide decision-makers on the urgency of adopting quantum-safe solutions.

To date, small-scale demonstrations of Shor's algorithm—such as factoring 21 [1,2]—have been successful but remain unscalable due to the absence of quantum error correction. Quantum systems are inherently unstable, and reliable error correction is essential for unlocking their full computational advantage.

A common misconception is that progress depends only on increasing qubit (the quantum version of a bit) counts. In practice, scalability hinges on reliability, error suppression, and coordination. Without this fault tolerance, more qubits just mean more noise. That's why achieving quantum error correction is fundamental—and the focus of this article.

Amid rapid technological progress, a communication gap has emerged: research is too technical for non-specialists, while media can sometimes exaggerate progress, which can result in misconceptions that overlook critical nuances. This article aims to bridge that gap with a clear framework for understanding quantum error correction, tailored specifically for cybersecurity professionals who are monitoring quantum developments. We construct and present building blocks for error-corrected quantum computers that will clarify the steps needed to realize a cryptographically relevant quantum computer.

What is error correction?

Computation relies on two core processes: storing and processing information—both of which are susceptible to errors. These errors can lead to incorrect results and need to be managed effectively in classical and quantum systems.

Classical computing uses error correction codes that add redundancy to detect and fix errors. These methods are so advanced that many users are unaware they're operating continuously.

Quantum error correction faces distinct challenges. Qubits are highly sensitive to noise, and quantum information cannot be copied (due to the no-cloning theorem [3]). Additionally, measuring a qubit might alter its state. These constraints make quantum error correction far more complex and resource-intensive than classical error correction.

The quantum error correction process

At the heart of quantum error correction lies the concept of encoding a single logical qubit into multiple physical qubits. A logical qubit represents a single unit of quantum information that is distributed across many physical qubits to make it more resistant to errors. By spreading the information this way, the system becomes less vulnerable to noise. This collective encoding is what allows quantum computers to suppress noise and correct faults.

Various quantum error correction codes exist, with surface codes and color codes being among the most widely used [4]. Two key parameters define such code:

- Overhead – the number of physical qubits required to encode one logical qubit.
- Distance – a measure of how many errors the code can detect and correct.

The ideal state would be high distance with low overhead, but design trade-offs often force a compromise. Choosing the applicable code depends on the architecture and the dominant noise sources.

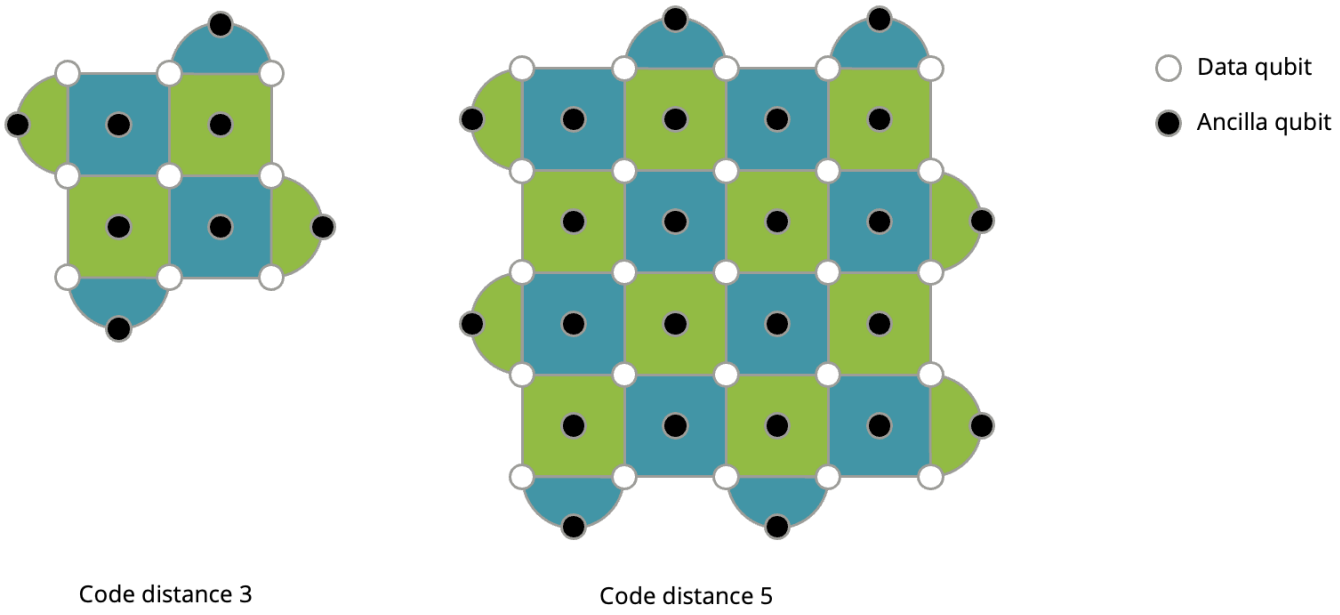

Figure 1: ***Surface code layouts for code distances 3 and 5.*** *This figure compares two surface code implementations. Each layout encodes a single logical qubit using a grid of data (white) and ancilla (black) qubits. Increasing the code distance—from 3 to 5—enhances error correction capability by allowing the code to detect and correct more errors, but requires significantly more physical qubits (17 for distance 3 and 49 for distance 5).*

For the chosen quantum error correction code, a separate group of auxiliary or ancilla qubits is used to perform continuous non-destructive measurements to extract potential errors and eventually to correct for them. Fig.1 shows an example of a distance 3 and distance 5 surface code, and depicts the data and ancilla qubits. The error correction process is depicted in Fig.2, and is composed of the following steps:

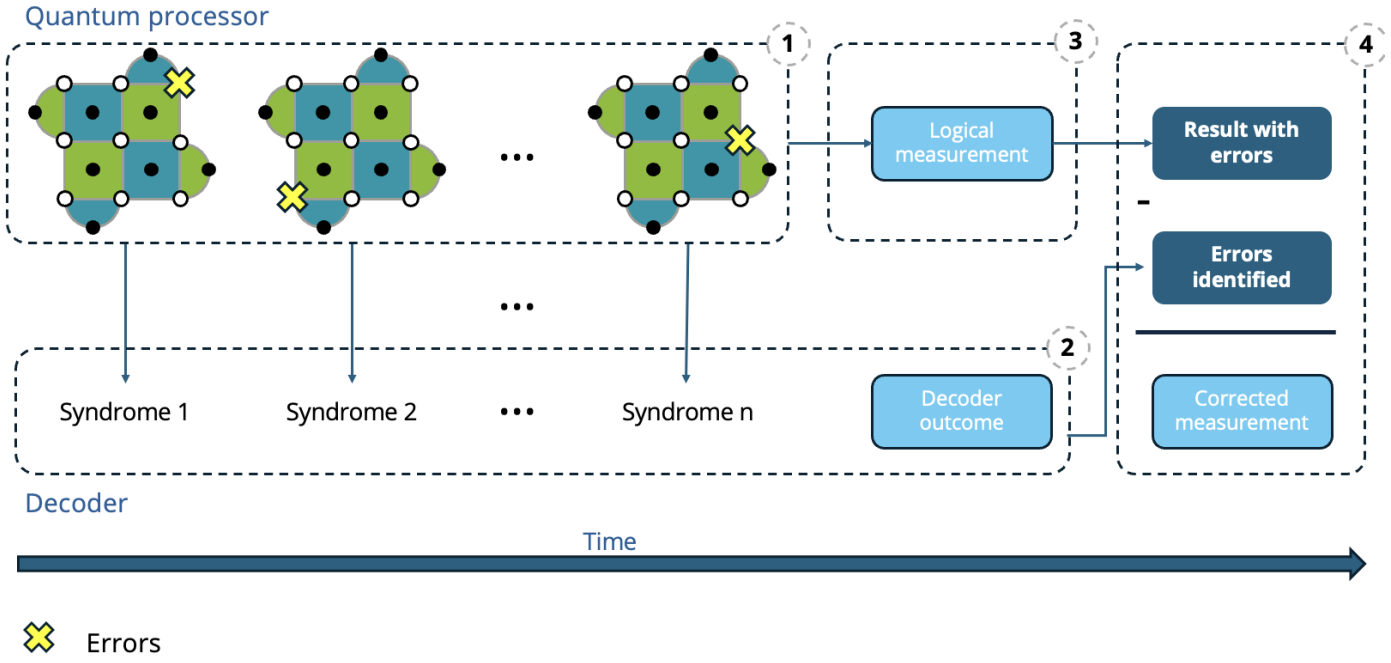

*Figure 2: **Key steps in the quantum error correction process**. This diagram illustrates the core components of quantum error correction: syndrome extraction, decoding, logical measurement, and correction. These steps work together to detect and mitigate errors in logical qubits, enabling fault-tolerant quantum computation—provided the physical error rate remains below a threshold.*

Syndrome extraction: This step entails performing a carefully defined set of entangling gates between the data qubits and the ancilla qubits, preserving the information encoded in the qubits. The state of the ancilla qubits is then measured, and the results, known as syndromes, provide insights into errors that may have affected the data qubit. Syndrome extraction occurs continuously throughout the operation of the quantum computer.

Decoding: While syndromes provide hints about potential errors, decoding infers the operation needed for correction. This process is computationally intensive. Decoding that to run continuously and at high speed is one of the challenges that needs to be overcome to support large scale quantum algorithms.

Logical measurements: At various points in the algorithm, it becomes appropriate to measure the state of the logical qubit. This measurement is carried out by measuring various combinations of data qubits.

Correction: When the decoding and logical measurement are done, relevant information to deduce what may likely have been the results of the measurement if no error had occurred is available.

If these steps are executed correctly, the resulting logical qubit will have a lower error rate than its underlying physical qubits. Importantly, this only holds if the physical system operates below a certain error threshold, which varies depending on the code, the noise model, and decoding accuracy.

Operating below threshold is a critical requirement for achieving fault tolerance. As we'll highlight in later sections, reaching and demonstrating below-threshold performance is one of the core milestones in building a scalable quantum computer.

Logical operations

Quantum algorithms that display an exponential speedup comparted to their classical counterparts require a universal gate set. One of the most widely used gate sets is Clifford+T [5], including:

- Clifford gates: Pauli (X, Y, Z), Hadamard (H), Controlled NOT (CNOT), and S
- Non-Clifford gate: T

Applying operations inevitably introduces new errors. Therefore, when logical gates are implemented, they need to be fault tolerant, which means that such errors can still be handled by the quantum error correction scheme.

As we delve deeper, we will see that Clifford gates are relatively easy to implement in a fault tolerant manner, while T gates are much harder to implement. As will be shown in the following section, implementing T gates correctly/properly is one of the

most challenging elements in realizing a fault tolerant quantum computer.

Building blocks for a fault-tolerant quantum computer

As outlined in the previous sections, numerous elements need to work in harmony to enable the execution of quantum algorithms in a fault-tolerant manner. Many of these components present experimental challenges and are often difficult to communicate in an accessible way to non-specialists.

We propose a structured framework to assess progress in fault-tolerant quantum computing. It has three levels, each with three building blocks (see Fig. 3). Full algorithmic capability depends on mastering each of them.

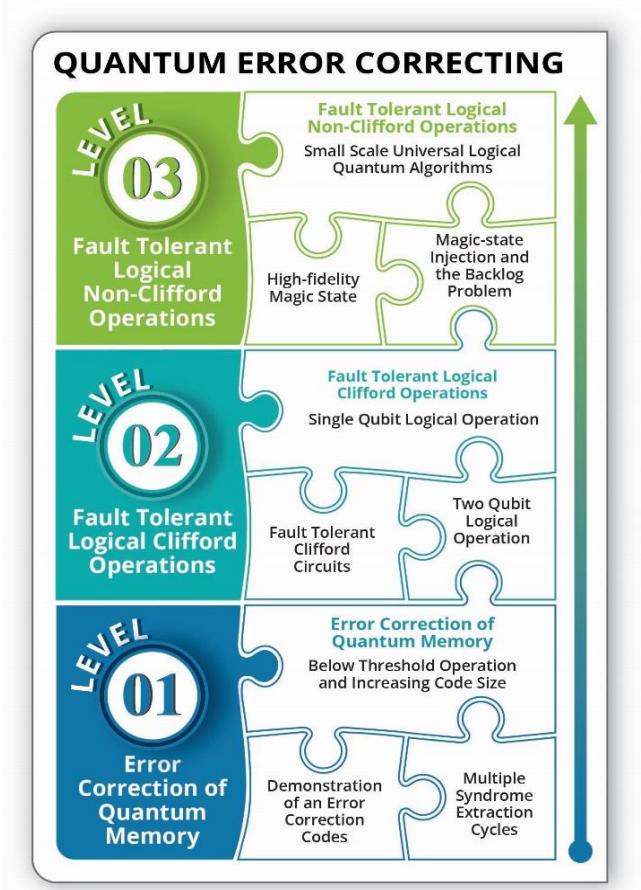

Figure 3: ***Three-level framework for fault-tolerant quantum computing.*** *This framework categorizes progress in fault-tolerant quantum computing into three levels: error correction of quantum memory, fault-tolerant logical Clifford operations, and fault-tolerant non-Clifford operations. Each level consists of three building blocks representing key capabilities. While presented in order, progress across levels can occur in parallel, depending on the quantum platform and experimental focus.*

Level 1: Error correction of quantum memory

At this level, the focus is on quantum error correction within quantum memory, without applying operations that alter information encoded in the logical qubit.

Building block 1: Logical qubit encoding and syndrome extraction

The first building block involves experimentally demonstrating the various elements of a quantum error correction code. This includes encoding data in a logical qubit, extracting syndromes, and decoding.

Early demonstrations of these fundamental properties began over 20 years ago and since then, many types of codes (such as repetition codes [6], surface codes [7], color codes [8]) have been tested. Initially, the logical error rates in these demonstrations were worse than the physical error rates. However, they primarily showcased the feasibility of these methods and identified areas for improvement.

Building block 2: Multiple syndrome extraction cycles

Due to the fragility of the qubits, syndrome extraction cycles need to be executed continuously throughout the runtime of the quantum algorithm. Such cycle can take anywhere from microseconds (for superconducting qubits) to milliseconds (for ions and neutral atom qubits), a significant number of syndrome extraction cycles are required.

Demonstrating multiple cycles necessitates improvements in the error rate of the physical qubits and addressing errors caused by other processes, such as measuring the ancilla qubits and applying physical gates. In recent years, several experiments have effectively demonstrated tens of syndrome extraction cycles and decoding [9,10,11,12].

Building block 3: Below-threshold operation and increasing code size

As described in an earlier section, increasing the code distance enhances the logical error rate only if the physical error rate remains below a certain threshold. In 2024, Google [13] provided a thorough demonstration of below-threshold operation. They experimentally established the exponential relationship between code distance and logical error rate. Specifically, they demonstrate that each increase in code distance improves the quality of the logical qubit by a factor of 2. Additionally, the team conducted up to 1 million syndrome extraction cycles, with an operational time of up to 1 second, marking a significant advancement from previous demonstrations.

Level 2: Fault tolerant logical Clifford operations

Building on the foundation established in Level 1, this level examines single and two-qubit Clifford operations, essential for executing quantum algorithms.

Building block 1: Single qubit logical operation

Various methods are available for implementing single qubit gates (Pauli gates, Hadamard, and S). These methods range from directly applying the physical gate to all physical qubits, to more advanced techniques like lattice surgery. Demonstrations of these building blocks can vary from proof-of-concept applications to embedding them in sequences of gates to illustrate a quantum algorithm. While implementing Pauli gates is relatively straightforward, the H and S gates may involve more complexity. A crucial aspect of these

demonstrations is controlling against adverse impact to the logical error rate from the incurred errors. Logical single qubit gates have been demonstrated in trapped ions [14,15] and neutral atoms [16] for different color codes and the surface code.

Building block 2: Two qubit logical operation

In our chosen universal gate set, the CNOT gate is the only two-qubit gate, playing a critical role in quantum algorithms through its function in creating entanglement. One way to implement a CNOT between two logical qubits is to perform pairwise physical CNOT gates between their constituent physical qubits. However, this requires connectivity between pairs of physical qubits that are far from each other. In qubit architectures where qubits can be moved (e.g., atoms and ions), the solution is to physically move the qubits closer to perform the pairwise CNOT. However, for architectures of fixed qubits (e.g., superconducting qubits), a CNOT gate can only be realized for qubits in close proximity to each other. Lattice surgery is a technique where the CNOT operation is executed by stitching qubits to each other and operating only on the interface (supporting the proximity limitation of fixed qubits).

Logical CNOT gates between color-code logical qubits have been demonstrated on both trapped-ion systems [17,18] and neutral-atom platforms [19]. Additionally, logical CNOTs between pairs of distance-7 surface-code qubits have been realized using neutral atoms [19]. Lattice surgery has also been used to entangle two distance-2 logical qubits in a trapped-ion quantum computer [20]. On superconducting platforms, lattice-surgery-based logical CNOTs have so far been implemented only for the repetition code [21].

Building block 3: Fault tolerant Clifford circuits

With the required components established, executing basic Clifford circuits can begin. This enables the application of the operations discussed at this level while correcting for errors that accumulate during the algorithm's runtime. Initial demonstrations of entanglement circuits [22] and simple quantum algorithms [23] have been made, though the number of syndrome extraction cycles in these experiments has been relatively low. To fully realize this building block, both the execution of logical operations and the demonstration of multiple syndrome extraction cycles have to occur concurrently.

While superconducting qubits have shown many syndrome extraction cycles (excelling in level 1), neutral atoms and trapped ions have demonstrated more versatile logical operations (level 2). It is yet to be seen which qubit modality will be the first one to demonstrate both level 1 and 2 simultaneously.

Level 3: Fault tolerant logical non-Clifford operations

Every quantum algorithm offering exponential speedups involves both Clifford and non-Clifford gates, such as the T gate. Most quantum error correction schemes natively support the fault-tolerant implementation of Clifford gates, while non-Clifford gates (in this case, the T gate) typically require indirect methods. The leading approach for implementing a T gate involves preparing a special type of quantum state—known as a magic state—which is then injected into the computation. This process is called magic state injection and is equivalent to applying a T gate in a fault tolerant manner.

Magic states are needed in large quantities, and their preparation consumes a substantial fraction of the qubits available in a quantum computer. Magic state injection is also a complex process, requiring additional qubits and fast decoding.

The building blocks required for implementing this process are as follows:

Building block 1 - High-fidelity magic state

High-quality magic states are essential for implementing fault-tolerant T gates. Typically, these states are produced through a process called magic state distillation, where multiple low-fidelity magic states are refined to achieve a target quality. Unfortunately, this process can consume the majority of the resources required for large-scale quantum computations. Nonetheless, recent advancements [24,25] have led to more efficient strategies for preparing magic states, aiming to reduce the overhead associated with distillation.

Interesting experiments in magic state distillation have been performed starting from physical qubits, rather than logical ones, in nuclear magnetic resonance quantum computers [26] and trapped ions [27]. Other experiments in trapped ions [28] and superconducting qubits [29,30] have demonstrated error-suppressed magic states using so-called flag protocols. While these are compelling demonstrations, they are difficult to scale. More recently, magic state distillation was demonstrated at the logical level on a neutral-atom quantum computer [31] using color codes and a 5-to-1 distillation protocol. This experiment observed improvements in the logical fidelity of the output magic states compared to the input logical states. As impressive as this result is, the researchers note themselves that a lot of work remains to be done before magic state distillation is performed at scale.

Building block 2 - Magic-state injection and the backlog problem

To apply a fault-tolerant T gate to a logical qubit, a three-step process known as magic state injection is used. First, a logical CNOT is performed between the magic state and the target logical qubit. Next, the modified magic state is measured logically, yielding 0 or 1. Based on this outcome, we determine whether a corrective phase shift is needed, implemented via a Clifford. After this process, the logical qubit's state is equivalent to having undergone a T gate. This indirect approach ensures that the operation is implemented in a fault-tolerant manner.

We wish to stress that injection involves logical branching on execution. The branching is conditioned on the result of a logical measurement. To trust the outcome of this measurement, the syndromes that precede it will need to be decoded. Therefore, quantum algorithms involving T gates require real-time error correction. Moreover, if decoding is not done swiftly enough, a syndrome backlog can accumulate, potentially leading to an exponential slowdown of the algorithm and negating the benefits of quantum computation. This last requirement calls for the

development of specialized decoding systems capable of supporting reliable quantum computation at scale.

There are proof-of-concept demonstrations of magic state injection in trapped ions [32], neutral atoms [33], and superconducting qubits [34]. These experiments are limited in scope, as proper implementation of magic state injection requires high-quality magic states and full enablement for level-two capabilities.

Building block 3 - Small scale universal logical quantum algorithms

A quantum computer that supports the previously outlined building blocks can run small-scale universal quantum algorithms in a fault-tolerant manner. A natural first step may be the fault-tolerant implementation of widely used algorithmic primitives. One such example is the Quantum Fourier Transform (QFT), a key subroutine in quantum algorithms for differential equations, chemical systems, and cryptanalysis. Executing a QFT involves both Clifford and non-Clifford operations, and its circuit depth necessitates numerous cycles of syndrome extraction.

The QFT was already demonstrated over 20 years ago using NMR systems [35]. This early experiment did not incorporate fault-tolerant techniques and was therefore not scalable to commercially relevant problems. It wasn't until 2024 that a partially fault-tolerant version was realized on a trapped-ion quantum computer [36]. Remarkably, the components required for a fully fault-tolerant implementation are still under development—showing just how challenging it is to build reliable quantum computers. Nonetheless, progress continues, and a fully fault-tolerant demonstration of QFT would mark a defining milestone in the field.

How to use this framework

This framework provides a structured lens for assessing progress in fault-tolerant quantum computing. Consider a hypothetical announcement of a 100 logical qubit hardware. Which algorithms can be executed on such a machine? To gauge the significance of such advancement, practitioners could pose critical questions such as:

1. What error correction code was employed? Understanding the specific code used allows for an assessment of the robustness of the error correction mechanism. Some codes are not scalable or cannot even correct certain types of errors (for instance the repetition code).
2. What was the logical error rate achieved, and how many syndrome extraction cycles can the system sustain? Compare physical and logical error rates to see if error correction works. Is below-threshold operation achieved? Can the system provide deeper circuits over longer runtimes?
3. Was a universal logical gate set implemented? Were both Clifford and non-Clifford gates executed in a fault-tolerant way? How was logical fidelity maintained across multiple syndrome extraction cycles?
4. What are the implications for scaling? Does the approach scale to larger qubit counts and deeper circuits, or are there architectural bottlenecks?

By asking these questions, practitioners can cut through technical complexity and media hype to judge how close a given experiment is to practical, scalable quantum computing.

Demonstrating a fault tolerant Shor's algorithm

With the framework established above, it is now possible to understand what it takes to factor integers in a fault-tolerant way—and what would have to happen before RSA is genuinely at risk. A recent paper [37] analyzed the requirements to factor the number 21 using a surface code implementation on superconducting qubits and many of the key ideas are accessible through the framework presented in this article.

The paper shows that factoring 21 fault-tolerantly requires 5 logical qubits with code distance 5 (i.e., 49 physical qubits per logical qubit), and around 200 rounds of syndrome extraction—executed in less than a millisecond. An additional ~750 physical qubits are needed for overhead associated with CNOT and T gates, with the T gates dominating the complexity.

While current hardware is not yet capable of meeting these requirements concurrently, the experimental progress referenced in this article—and aggressive roadmaps from major quantum players—suggest a demonstration of this scale is plausible in the next few years. It would mark a watershed moment: the first truly fault-tolerant implementation of Shor's algorithm.

For contrast, a 2025 study [38] estimated the resources needed to factor a 2048-bit RSA number using the same surface code on superconducting qubits. The comparison below (Table 1) illustrates how the jump from factoring 21 to factoring 2048-bit integers requires both scaling and improvement at each level of the stack.

Notably, while the hardware gap between these two estimates is vast, researchers are making continuous improvements in reducing resource requirements. Early estimates from 2009 [39] projected that factoring a 2048-bit RSA number may likely require 6.5 billion physical qubits and 410 days of runtime. In [38], that estimate was reduced to below 1 million qubits — a dramatic reduction by multiple orders of magnitude, due to advances in error correction, compiler optimization, and circuit design.

As hardware capabilities improve and algorithmic requirements continue to shrink, these two trajectories are converging. This reinforces the urgency to mitigate the quantum risk.

	Factoring 21	Factoring 2048 bit numbers
Logical qubits	5	1,399
Physical qubits	1,015	<1M
Code distance	5	25-30
Runtime	200 µs (195 cycles)	~1 week
T gates	13	>10 billion

Table 1: Comparison of resource requirements for fault-tolerant implementations of Shor's algorithm.

This table contrasts the estimated qubit counts, code distances, runtimes, and T gate requirements for factoring the number 21 versus a 2048-bit RSA number using surface code error correction. It illustrates the gap between near-term demonstrations and cryptographically relevant quantum attacks.

Conclusion

Quantum computing has made remarkable strides in recent years. Experiments once considered theoretical milestones—like below-threshold operation, logical gate execution, and even partial implementations of fault-tolerant quantum algorithms—are now becoming reality. Yet, the path to running cryptographically relevant algorithms like Shor's remains challenging.

As this article outlined, progress depends not just on qubit counts but on mastering a full stack of fault-tolerant capabilities. These include reliable quantum memory, universal logical gates, and real-time error correction—each of which are essential for executing meaningful quantum algorithms at scale.

The framework introduced here provides a clear way to evaluate these developments and communicate them without oversimplification or hype. It helps decision-makers distinguish between proof-of-concept experiments and genuine breakthroughs with cybersecurity implications.

Researchers, cybersecurity professionals, and the media can use this framework when assessing or reporting on quantum advancements. It offers a grounded, structured way to cut through ambiguity and understand how close the practical quantum advantage really is—and what is still to be done to get there.

The core takeaway: the industry is not there yet—but progress is being made, faster than many expected. Understanding where the industry is along this trajectory is essential to prepare for what comes next. Waiting for a fully fault-tolerant quantum computer before acting on the quantum risk may be too late.

References

[1] Skosana, U., Tame, M., Demonstration of Shor's factoring algorithm for N 21 on IBM quantum processors. Sci Rep. 2021.

[2] Monz, T. et al., Realization of a scalable Shor algorithm. Science. 2016.

[3] Wootters, W. & Zurek, W., A single quantum cannot be cloned. Nature. 1982.

[4] Albert, V., Faist, P., & Contributors., List of code lists. Error Correction Zoo. n.d.

[5] Gidney, C., Shutty, N., & Jones, C., Magic state cultivation: growing T states as cheap as CNOT gates. 2024.

[6] Kelly, J., Barends, R., Fowler, A., et al. State preservation by repetitive error detection in a superconducting quantum circuit. Nature. 2015.

[7] Marques, J., Varbanov, B., Moreira, M. et al., Logical-qubit operations in an error-detecting surface code. Nat. Phys. 2022.

[8] C. Ryan-Anderson, et al., Realization of Real-Time Fault-Tolerant Quantum Error Correction. 2021.

[9] Andersen, C.K., Remm, A., Lazar, S. et al., Repeated quantum error detection in a surface code. Nat. Phys. 2020.

[10] Krinner, S., Lacroix, N., Remm, A. et al., Realizing repeated quantum error correction in a distance-three surface code. Nature. 2022.

[11]Sundaresan, N., Yoder, T.J., Kim, Y. et al., Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders. Nat Commun. 2023.

[12] Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. Nature. 2023.

[13] Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. Nature. 2025.

[14] Ryan-Anderson, C. et al., Realization of Real-Time Fault-Tolerant Quantum Error Correction. 2021.

[15] Wang, Y. et al., Fault-tolerant one-bit addition with the smallest interesting color code. Science Advances. 2024.

[16] Bluestein, D., Evered, S., Geim, A. et al., Logical quantum processor based on reconfigurable atom arrays. Nature. 2024.

[17] Ryan-Anderson, C., Implementing fault-tolerant entangling gates on the five-qubit code and the color code. arXiv. 2022.

[18] Postler, L., Heußen, S., Pogorelov, I. et al., Demonstration of fault-tolerant universal quantum gate operations. Nature. 2022.

[19] Blustein, D., Evered, S.J., Geim, A.A. et al., Logical quantum processor based on reconfigurable atom arrays. Nature. 2024.

[20] Erhard, A., Poulsen-Nautrup, H., Meth, M. et al., Entangling logical qubits with lattice surgery. Nature. 2021.

[21] Besedin, I. et al., Realizing Lattice Surgery on Two Distance-Three Repetition Codes with Superconducting Qubits. 2025.

[22] Bluestein, D., Evered, S.J., Geim, A.A. et al., Logical quantum processor based on reconfigurable atom arrays. Nature.. 2024.

[23] Reichardt, B. et al., Logical computation demonstrated with a neutral atom quantum processor, Arxiv. 2024.

[24] Litinski, D., Magic State Distillation: Not as Costly as You Think. Quantum. 2019.

[25] Gidney, C., Shutty N., & Jones, C., Magic state cultivation: growing T states as cheap as CNOT gates. Arxiv. 2024.

[26] Souza, A., Zhang, J., Ryan, C. et al., Experimental magic state distillation for fault-tolerant quantum computing. Nat Commun. 2011.

[27] Brown, N., et al., Advances in compilation for quantum hardware -- A demonstration of magic state distillation and repeat-until-success protocols. Arxiv. 2023.

[28] Postler, L., Heuβen, S., Pogorelov, I. et al., Demonstration of fault-tolerant universal quantum gate operations. Nature. 2022.

[29] Ye, Y., Logical Magic State Preparation with Fidelity beyond the Distillation Threshold on a Superconducting Quantum Processor. APS. 2023.

[30] Gupta, R.S., Sundaresan, N., Alexander, T. et al., Encoding a magic state with beyond break-even fidelity. Nature. 2024.

[31] Rodriguez, P., Experimental Demonstration of Logical Magic State Distillation. Arrive. 2024.

[32] Ryan-Anderson, C., Realization of Real-Time Fault-Tolerant Quantum Error Correction. APS. 2021.

[33] Postler, L., Heuβen, S., Pogorelov, I. et al., Demonstration of fault-tolerant universal quantum gate operations. Nature. 2022.

[34] Kim, Y., Sevior, M., & Usman, M., Magic State Injection on IBM Quantum Processors Above the Distillation Threshold. Arxiv. 2024.

[35] Weinstein, Y.S. et al., Implementation of the Quantum Fourier Transform. APS. 2001.

[36] Tao, R., Enhancing Real-World Active Speaker Detection with Multi-Modal Extraction Pre-Training. Arxiv. 2024.

[37] Kurman, Y., Controller-decoder system requirements derived by implementing Shor's algorithm with surface code. Arxiv. 2024.

[38] Gidney, How to factor 2048 bit RSA integers with less than a million noisy qubits. Arxiv. 2025.

[39] Van Meter, R. et al., Distributed Quantum Computation Architecture Using Semiconductor Nanophotonics. ArXiv. 2009

Deloitte.