Reassessing Grover's Algorithm

Scott Fluhrer

Cisco Systems, USA
sfluhrer@cisco.com

Abstract. We note that Grover's algorithm (and any other quantum algorithm that does a search using an oracle) does not parallelize well. Accordingly, we propose a modified security assumption, that the attacker has bounded time to perform the attack in addition to an overall computational budget. We show that, under this security assumption, the size of the problems that Grover's algorithm can attack is less than commonly assumed. For example, we show that for symmetric keys, we don't need to double their size, adding a fixed number of bits is sufficient. This reduction in strength can be used to make postquantum cryptography to be of lesser cost, without sacrificing security.

1 Introduction

Postquantum cryptographical algorithms are generally more expensive than their corresponding non-postquantum equivalents. This additional expense is due to the need to be infeasible to attack by quantum algorithms; it seems reasonable to assess the actual strength of these algorithms.

In 1996, Grover introduced his well publicized algorithm to do database searches [8], which can search through a space of size N in $O(\sqrt{N})$ time. Here, the attacker is searching for a database entry (or cryptographical key) that is distinguished; it performs successive evaluations of the Oracle being searched over an entangled state, interleaved with relatively simple quantum operations on that state; after about $\pi/4\sqrt{N}$ such iterations, a measurement of that state will give the value being searched for with high probability.

Because of this algorithm, it has been commonly accepted that (for example) we need to double the size of our symmetric keys in order to maintain the security level against a Quantum Computer. For example, to obtain "128 bit quantum security", that is, require the attacker to perform circa 2^{128} quantum operations to successfully attack the system, we need a symmetric cipher with a 256 bit key.

And, yes, if the attacker has a Quantum Computer that can issue circa 2^{128} successive evaluations to our encryption function, they could find the key with high probability. However, in this attack model, even a 160 bit key (which would require circa 2^{80} successive evaluations), would be secure for all practical purposes. Even if we assume that the quantum computer could evaluate the encryption function in the absurdly fast time of 1 picosecond, the total of 2^{80} successive evaluations would require over 30,000 years to complete. Any realistic security policy is not concerned with an attack that takes that long to perform.

The reason we insist on 128 bit security is because of parallelization; we can divide the search task over a large number of parallel processors, each with a separate implementation of the encryption function oracle, which can be queried in parallel. On a conventional computer, this search process is arbitrarily parallelizable; if we have a single oracle, then with T queries, we can search through a space of size $O(T)$, if we're given S Oracles which we can query in parallel, then we can perform a total of ST queries, and are able to search a space of size $O(ST)$.

2 Parallelization of Grover's Algorithm

We would like to consider the corresponding parallelizability on a Quantum Computer; that is, given a parallel Quantum Computer, with access to a number of oracles, how fast can they perform a search for a distinguished input. It turns out that the limits of what can be achieved by any such quantum algorithm has been proven in [12]. This paper is well known for showing that Grover's algorithm is an optimal solution to this problem given a single oracle; however, this paper also considers the scenario if the Quantum Computer is given a number of oracles in parallel that it can query simultaneously (see section [4]).

Specifically, Zalka proves that any quantum algorithm which does a search based on issuing entangled oracle queries necessarily has

$$2N - 2\sqrt{N}\sqrt{p} - 2\sqrt{N}\sqrt{N-1}\sqrt{1-p} \leq S \cdot 4T^2$$

(see equations (8), (26) from the paper), where:

- N is the size of the space being searched over
- S is the number of parallel oracles available to the attacker
- T is the number of times he can issue queries to eachoOracle in parallel
- p is the success probability of deriving the correct key

The paper further notes that this inequality is tight; equality can be achieved by splitting up the search space into S equal sized subregions, and have a separate implementation of Grover's algorithm search each subspace.

This paper explicitly assumes that the oracle has a unique solution; that is, there will be exactly one value that will cause the oracle to evaluate to the searched for result. For some of the uses we will use this theorem, we will need to handle the general case where there are t distinct values that will satisfy the Oracle query, and the search will be successful if we find any of them[1].

It is easy to extend Zalka's result for general t; we get:

$$2N - 2\sqrt{Np/t} - 2\sqrt{N(N-1)(1-p/t)} \leq S \cdot 4T^2$$

[1] [5] gives a Quantum algorithm for evaluating t for an oracle; this is important for the original Grover's algorithm; however as long as $t \ll S$, the exact value of t is unimportant to the attack which uses S separate Grover's searches.

We outline how to modify Zalka's proof in appendex 1.

We note that

$$2N - 2\sqrt{Np/t} - 2\sqrt{N \cdot (N-1) \cdot (1-p/t)} \approx 2N(1 - \sqrt{1-p/t}) > Np/t$$

(where the approximation holds for $p \gg t/N$) and so we have:

$$Np/t < 4ST^2$$

If we denote the total amount of Oracle calls that the adversary can make as $C = pST$, then we have $N/t < 4CT$.

3 Proposed Security Assumptions

Because of this, we propose that we modify the security assumptions that we make to include the factor of time. In particular, we would make three explicit assumptions.

- The first assumption is one of total computational effort pC, where C is the overall computational budget, and p is the minimum success probability that the adversy will tolerate. For example, in the 128 bit security case, to obtain a success probability of p, he must involve the Oracle at least $2^{128}p$ times. This is the standard computational complexity assumption.
- The second assumption is one of time T_1; we assume that the attacker must complete his attack within a certain timeframe; for example, the attack must be completed within 200 years.
- The third assumption is of speed; we assume that the Quantum Computer cannot issue a query to the Oracle and get a response back within a certain amount of time T_0; say, within 3nsec.

We allow the attacker to use arbitrary amounts of parallelization, as long as he stays within the total computational assumption of no more than pC queries.

Against an adversary with a conventional computer, these additional assumptions do not make the attacker's problem any harder; we allow him to increase the amount of parallelization high enough to perform the search in the given amount of time.

However, against an adversary with a Quantum Computer, these additional assumptions are nontrivial; if we denote $T = T_1/T_0$ (the total number of successive Oracle queries possible in the time allowed), then the attacker can succeed with probability p only if:

$$\log_2 N/t < log_2\, C + (2 + \log_2 T)$$

That is, to reach a given security level against Quantum Computers using Grover's algorithm (or any other quantum oracle search algorithm), we don't need to double the size of the symmetric keys; it suffices to add a fixed number of bits (the number of bits depends on how fast we assume the Quantum Computer is compared to the time we allow the attacker).

3.1 Reasonability of Security Assumptions

We are adding two security assumptions; the obvious questions are whether those assumptions are reasonable, and if so, what are appropriate values for T_0, T_1.

As for the time bound, it is common for security policies to have a time bound; there is no data that needs to be kept private literally forever, and for a signature scheme, no public key will be used forever. It is unclear how long the time bound could be; we would tentative recommend a time bound T_0 of 200 years (as it is hard to think of anything that needs to be kept private that long).

As for the speed bound, that is trickier. We are attempting to make a bound on how fast a Quantum Computer will perform, and we don't have a practical Quantum Computer[2] in front of us.

However, we can look at the existing state of the art and extrapolate from there; we will take as example cryptographical Oracles the SHA-256 and the AES-256 functions, as they have been studied.

In [2],a quantum circuit for implementing the SHA-256 hash function is proposed, along with the estimate of the cost (including the total quantum gate delay); their optimized implementation gives a T-Depth[3] of 70400[4].

In [7], a quantum circuit for implementing the AES-256 encryption function is proposed; their implementation gives a T-Depth of 7488.[5]

The current model is that, after each gate, we will need to implement Quantum Error Correction, which involves measuring physical qbits which are set to be a function of the actual data physical qbits. The current fastest we can perform such a measurement is in 48nsec, as in [11]

Assuming that the measurement time takes the bulk of the operation time, then this gives us a time of 3.38msec to implement the SHA-256 hashing function on a quantum computer, and 0.359msec to implement the AES-256 function.

Now, these estimates are based on the current state of the art; quite likely, future researchers will find ways to improve these values; by providing alternative implementations with less depth; by reducing the decoherence probability (so we don't need to implement error correction as frequently), and by speeding up the measurement process. Because of these uncertainties, I would suggest that we derate these parameters by a factor of a million; giving us a $T_1 = 3$nsec for SHA-256 and $T_1 = 0.3$nsec for AES[6].

[2] That is, one that is capable of attacking real cryptographic problems.

[3] T-Depth [1] is the number of stages in the quantum circuit involving non-Clifford group gates. This is a metric commonly used in quantum computing, as these gates are expected to be the bottleneck in fault-tolerant computation.

[4] They also give a proposed Quantum implementation of SHA-3; that has a signfiicantly shallower T-depth of 432. It is not immediately clear if this indicates that SHA3 has somewhat less resistance to quantum attacks, or whether that's an artifact of this analysis, which considers implementation speed, but not implementation cost.

[5] The smaller depth for AES reflects the greater parallelizability inherent within the AES structure.

[6] Both these speed estimates are approximately an order of magnitude faster than how fast we can implment these functions using conventional gates.

When we combine these two estimates, we obtain $T < 2^{61}$ for SHA-256, and $T < 2^{65}$ for AES.

One can argue that these values might not be conservative enough; however any reasonable value of T_0, T_1 allows some improvement over the traditional view of Grover's algorithm.

4 Implications of These Security Assumptions

Now, if you go through this argument, you'll find that this implies that AES-192 has (with the above assumptions) almost 128 bits security; it would take circa 2^{125} entangled decryptions to recover an AES-192 key. However, we are not suggesting that people abandon AES-256 in favor of AES-192. By this argument, AES-256 has far more than "128 bits quantum security", by the above security assumptions, it has about 190 bits strength, however the addiitonal cost of AES-256 is minimal, and so there is little to be gained by switching to AES-192.

This is not true for other operations, which can become significantly less expensive if we adjust the security parmeters in accordance to these security assumptions.

4.1 Hash Based Signatures

A hash based signature is a signature method whose cryptographical strength relies solely on the strength of the underlying hash function; if we believe that the hash function is strong (e.g. is first and second preimage resistant), then it is infeasible to generate forgeries. Proposed hash based signatures include XMSS [9], LMS [10] and Sphincs [4]. We will focus on Sphincs as that scheme has the fewest practical issues; similar size reductions occur with these other schemes as well.

The Sphincs structure consists of twelve levels of Merkle trees, where each tree uses a one time signature (WOTS+) to sign the root of the tree immediately below it; each Merkle tree is of height 5, and so we have hypertree with 2^{60} leaf nodes. Each node can be used to sign the root of a HORST tree, which is a few time signature method (that is, one which can be used to sign a few messages). To generate a Sphincs signature, we hash the message (and a random value) to select one of the leaf nodes of the hypertree; we then form the HORST tree that corresponds to that leaf node, and generate the HORST signature of the message, and the authentication path through the hypertree to the top level root value (which is the public key).

Against such a primitive, we expect that oracle queries are the best possible attack[7], and hence the quantum oracle search is what we are concerned with.

Against Sphincs, the best attack strategy would be to attack either one of the unrevealed HORST leafs (adjacent to one of the revealed ones), or the previous value in the WOTS+ chain. In both cases, these are preimage attacks where we

[7] If the hash function is such that it allows a better attack, we should replace it with a better hash function.

know (by construction) there is at least one preimage (the one that the legitimate signer would reveal); hence there are an expected 2 such preimages (and hence $t = 2$).

We would also note that, as a public key signature method, the timespan for the attacker has for any forgery to be useful must be within the lifetime of the public key; that is, when someone with the public key is still willing to accept a message verified by it. In [3], NIST recommends that public key cryptoperiods be limited to "on the order of several years"; one could argue that our value $T_0 = 200$ years could be reduced. However, for this analysis, we will retain that value.

If we use the security assumptions given above[8], we find that a 192 bit hash function[9] provides a full "128 bits Quantum Security"; and hence meets the original Sphincs security target.

The original Sphincs signature method has signatures which are 41000 bytes long; however, we can take advatage of our security assumptions to shrink the signatures in three ways:

– The Sphincs signature consists largely of a series of hashes; if each hash is 192 bits rather than 256 bits, each hash contained in the signature is 8 bytes smaller.
– The signanture includes a number of WOTS+ signatures with $W = 16$; to sign a 256 bit hash, each WOTS+ signature consists of 67 hashes. By making each signed hash be only 192 bits, each WOTS+ signature would now consist of 51 hashes.
– The HORST signature consists of $k = 32$ authentication chains from the bottom of the Horst tree. They selected $k = 32$ because that implies that if a HORST tree was used $\gamma = 8$ times, then a distinct message that the attacker selects would have probability $\leq 2^{-256}$ of consisting of already revealed leaf values. This probability of 2^{-256} was selected because Grover's algorithm can be used to select such a message; however we can show that a probability of 2^{-192} is sufficient to be Quantum Secure; this allows us to use a value $k = 23$, which implies that we need fewer authentication chains in the signature.

As a result of these three observations, we can safely shrink the Sphincs signature to about 23k, while retaining all of the security goals. We also improve the signature generation time somewhat (because computing the internal WOTS+ public keys becomes cheaper), but not as dramatically.

5 Conclusion

We have shown that Grover's algorithm, in practice, is less powerful than generally assumed. Many postquantum primitives can be criticized as expensive; for some

[8] Sphincs uses Blake as its internal hash function; for this analysis, we will assume that either a value $T_1 = 3$ nsec is appropriate for Blake, or alternatively we replace it with SHA-256.

[9] Which can be implemented by computing our 256 bit hash function, and just using the initial 192 bits.

of them, specifically, the ones for which a Grover search is the best attack, this observation may allow us to safely reduce the security parameters, and thus reduce the cost, while staying within the design security bounds.

We have made suggestions for values for T_0, T_1; while we feel that our value for T_1 is safe, it could be criticized as being both arbitrary and not conservative enough. We would still recommend that this approach still be considered; a quamtum computer will require some amount of time to compute a cryptographical function, and with a smaller T_1, this analysis gives us some (albeit less) practical benefit.

I would like to thank David McGrew and Andreas Hülsing for their useful comments, and I would especially like to thank John Schanck with help with my questions on Quantum Computation speed.

6 Future Work

Open questions that may be addressed in future work:

– Can we obtain a better estimate on what the value T_1 should be?
– What other postquantum cryptographical primitives can be optimized using this observation?
– We used existing analysis for our example cryptographical functions, AES-256 and SHA-256. There may be no corresponding analysis for the oracles that is queried during an attack against other primitives, such as a lattice-based primitive. How should we proceed with the analysis in that case?

References

1. Matthew Amy, Dmitri Maslov, Michele Mosca, Martin Roetteler *A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems https://arxiv.org/abs/1206.0758
2. Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, John Schanck *Estimating the cost of generic quantum pre-image attcks on SHA-2 and SHA-3* Selected Areas in Cryptography 2016 https://eprint.iacr.org/2016/992.pdf
3. Elaine Barker *Recommendation for Key Management; Part 1: General* http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4
4. Daniel J. Bernstein, Daria Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Peter Schwabe, Zooko Wilbox O'Hearn *SPHINCS: practical stateless hash-based signatures* https://cryptojedi.org/papers/sphincs-20141001.pdf
5. Michel Boyer, Gilles Brassard, Peter Høyer, Alain Tapp *Tight bounds on quantum searching* PhysComp96 https://arxiv.org/abs/quant-ph/9605034
6. Austin G. Fowler, Matteo Mariantoni, John M. Martins, Andrew N. Cleland *Surface Codes: Towards practical large-scale quantum computation* Phys. Rev A 86, 03324 (2012) https://arxiv.org/abs/1208.0928

7. Markus Grassl, Brandom Langenberg, Martin Roettler, Rainer Steainwandt *Applying Grover's algorithm to AES: quantum resource estimates* PQCrypto 2016 https://arxiv.org/pdf/1512.04965.pdf
8. Lov K. Grover, *A fast quantum mechanical algorithm for database search*, STOC 1996, https://www.arxiv.org/abs/quant-ph/9605043
9. A. Hülsing, D. Butin, S. Gazdag, J. Rijneveld, A. Mohaisen *XMSS: Extended Hash-Based Signatures* https://datatracker.ietf.org/doc/draft-irtf-cfrg-xmss-hash-based-signatures/
10. D. McGrew, M. Curcio, S. Fluhrer *Hash-Based Signatures* https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs/
11. T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočnik, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, A. Wallraff *Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits* Phys. Rev. Applied, vol. 7, issue 5 https://journals.aps.org/prapplied/abstract/10.1103/PhysRevApplied.7.054020
12. Christof Zalka *Grover's quantum searching algorithm is optimal*, Phys.Rev. A60 (1999) 2746-2751, https://arxiv.org/abs/quant-ph/9711070

7 Appendix 1: Parallel Oracle Searches with Multiple Matching Targets

Let us consider the case where we have a Quantum Computer with S identical oracles, however instead of having one marked elements that we are searching for, we have t such elements.

Zalka's proof consists of two parts, the first where he proves the inequality (26) and the second where he proves the inequality (8).

First, we can easily see that, in Zalka's [12] proof, his inequality (26)

$$\sum_y |\phi_T - \phi_T^y|^2 \leq S \cdot 4T^2$$

is not modified by $t > 1$; the proof of this deals with the difference between the quantum state in the case of an oracle with t elements that answer 'yes', and the case of an oracle that always answers 'no', that difference is not affected by the value t.

As for equation (8)

$$2N - 2\sqrt{N}\sqrt{p} - 2\sqrt{N}\sqrt{N-1}\sqrt{1-p} \leq \sum_{y=0}^{N-1} |\phi_T^y - \phi_T|^2$$

We can see how this is modified by going through the derivation of this inequality in section 5.1.2. We redefine ϕ_y[10] in equation (29) to be the quantum state when we have an oracle that gives yes for t distinct answers (while ϕ still designates the state when we have an oracle that gives a no answer for all inputs).

We find that the remainder of the proof applies unchanged until we get to the constraint $1/N \sum_y p_y = p$ found after equation (35); with our modified

[10] In his appendix, Zalka shifts the designation of the quantum state from ϕ to ψ; for consistency, I will continue to denote the state as ϕ.

assumption, this constraint becomes $t/N \sum_y p_y = p$. The other constraints remain unchanged, and so the Lagrange multiplier technique gives us $a_y = 1/N\ \forall y$ and $p_y = p/t\ \forall y$.

This then gives us the modified formula:

$$\sum_{y=0}^{N-1} |\phi_y - \phi|^2 \geq 2N - \sqrt{Np/t} - 2\sqrt{N(N-1)(1-p/t)}$$

This is the formula we have used above.