Basics of Quantum Algorithms: Grover's Algorithm

Tuur Willio 2073927

Technische Universiteit Eindhoven, TU/e

Abstract. Explanation of Grover's algorithm and how it could be used to attack symmetric key algorithms and hash functions.

1 Problem Introduction

Grover's algorithm performs an *unstructured* search [15] on a wanted item $\bar{x}$ in a scrambled dataset of size $|\Sigma^n| = N = 2^n$, where n is the number of qubits (introduced in section 2). This means that we have a set of data that has no structure that will aid in finding that certain item $\bar{x}$.
To locate that item, we use an oracle f:

$$f : \Sigma^n \longrightarrow \{0,1\} : x \longmapsto \begin{cases} 0\,, x \neq \bar{x} \\ 1\,, x = \bar{x} \end{cases}$$

where Σ^n is the 'alphabet' containing all items.The best way to approach this question, without quantum computing, is brute forcing. We will assume that the function f can be evaluated in constant time, $\mathcal{O}(1)$. Hence, performing a brute force attack will result in a time complexity of $\mathcal{O}(N) = \mathcal{O}(2^n)$. Depending on the size of the data, this is not very optimal. Grover's algorithm will find the correct item, with probability $\frac{1}{2}$, in time complexity $\mathcal{O}(\sqrt{N})$. This algorithm can be repeated to increase chances.

2 Introduction to Quantum Computing

This is a short explanation of the concepts and tools that are necessary in the algorithm or useful to understand the mechanics.

Quantum computing uses quantum mechanics for computing/algorithmic problems. A classical computer consists of bits, each able to represent either one or zero. This implies that, for example, in unstructured search problems, as described in section 1, a brute-force attack will have to evaluate each option separately. Note that brute forcing takes immense amounts of time and is an important characteristic of cryptographic systems, see section 4.
This is where quantum computing is different from classical computers. We do not talk about bits any more, but about *qubits* (quantum bits).

2.1 Qubits

Qubits are quantum states in superposition, represented by classical binary states $|0\rangle$ and $|1\rangle$ ('ket' notation). A superposition can be expressed as a linear combination of classical states:

$$|\varphi\rangle = \sum_{i=0}^{N} \alpha_i \, |i\rangle$$

where α_i are amplitudes, with $|\alpha_i|^2$ the probability of observing that specific state $|i\rangle$. For any quantum state $|\varphi\rangle$, the sum of the probabilities must equal 1: $\sum_{i=0}^{N} |\alpha_i|^2 = 1$, by law of total probability. Unlike

classical probabilistic computing, quantum amplitudes can be complex or negative. Like the famous thought experiment of Schrödinger's cat[14], we do not know in what state, dead or alive, the cat is unless we observe it. When we observe a superposition, it is 'dropped' into that classical state with a probability $|\alpha_i|^2$. The superposition is no more after being observed. More intuitive, a quantum state can be represented by a vector. For example, $|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, the superposition of one and zero with equal probability, can be represented by vector:

$$|+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad \begin{matrix} \sim |0\rangle \\ \sim |1\rangle \end{matrix}$$

The classical bit states $|0\rangle$ and $|1\rangle$ can then be represented by $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$ respectively. Similarly, $\langle 0|$ (note the notation difference with $|0\rangle$) is the transpose and is thus a row vector (called 'bra'). The 'bra' and 'kets', read bra-kets, notation is now clearly derived from the inner product notation of a measure space.The qubits will satisfy $\langle\varphi|\varphi\rangle = \langle\varphi| \cdot |\varphi\rangle = ||\varphi||^2 = 1$ (total law of probability). Vectors $|0\rangle$ and $|1\rangle$ are also orthogonal, their inner product is zero, and thus orthonormal. Qubits can also be represented on the Bloch sphere [9]. This is a three-dimensional sphere of radius one, with $|0\rangle$ and $|1\rangle$ on the poles of the sphere. In a system of multiple qubits, the classical states are represented by a tensor product, the kronecker product, of $|0\rangle$ and $|1\rangle$. For example, with two qubits, the states are represented by

- $|0\rangle \otimes |0\rangle =: |00\rangle$
- $|0\rangle \otimes |1\rangle =: |01\rangle$
- $|1\rangle \otimes |0\rangle =: |10\rangle$
- $|1\rangle \otimes |1\rangle =: |11\rangle$

. Notation of $|b_1 b_2\rangle$ is equivalent with value b_1 being observed in the first qubit, and b_2 in the second, $b_{1,2} \in \{0, 1\}$. Depending on the number of qubits n, we will be able to represent basic states zero to $N - 1 = 2^n - 1$ in a superposition.

2.2 Gates

Next to measuring, we can also apply operators on qubit systems. These operations are also representable by matrices. They are also unitary, this means they preserve the norm, $|| A|\varphi\rangle || = || |\varphi\rangle || = 1$, which is necessary by the total law of probability. We will use these operators, also called gates;

- Hadamard gate:

$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad H_{2^k} = H \otimes H_{2^{k-1}} = \frac{1}{\sqrt{2}}\begin{bmatrix}H_{2^{k-1}} & H_{2^{k-1}}\\ H_{2^{k-1}} & -H_{2^{k-1}}\end{bmatrix}$$

  This gate is great for initialisation. Apply $|0\rangle$, for example, to this gate: $H|0\rangle = \frac{1}{\sqrt{2}}[1, 1]^T =: |+\rangle$. Now both states are equally likely to be observed. This works the other way around as well, $H|+\rangle = |0\rangle$. In the general dimension case, this is a matrix containing only 1 and -1 with scalar: $\frac{1}{2^{n/2}}$, with $N = 2^n$[2]. An element in the Hadamard matrix looks like $(H_N)_{ij} = \frac{1}{\sqrt{2^n}}(-1)^{\bar{i}\cdot\bar{j}}$, with $\bar{i} \cdot \bar{j}$ being the bitwise dot product. Hadamard is also unitary, because $H^2 = I_N$ with I_N the identity matrix. Hadamard is its own conjugate transpose, because of the symmetry in its definition and non-complex values. This results in $H^*H = I_n$, with H^* the conjugate transpose. This is equivalent with unitary[5].

- Rotation gate:

$$R_4 = \begin{bmatrix} e^{i\theta_1} & 0 & 0 & 0 \\ 0 & e^{i\theta_2} & 0 & 0 \\ 0 & 0 & e^{i\theta_3} & 0 \\ 0 & 0 & 0 & e^{i\theta_4} \end{bmatrix} \qquad \text{general case, } (R_{2^n})_{jk} = \begin{cases} e^{i\theta_j} & k = j \\ 0 & k \neq j \end{cases}$$

Here is $i = \sqrt{-1}$ the complex unit. This performs as expected, a rotation on the states. The rotation of each state does not change the probabilities because $|e^{i\theta}| = 1$. For this reason, it is also unitary, $||Av|| = 1$ if $||v|| = 1$ [5].

3 The Algorithm

The essence of the algorithm is that it flips the sign of the sought after item using the oracle f, and than does an *inversion over the average*[10], which will amplify the state linked with our wanted item $\bar{x}$. And thus making its probability to be measured higher.

1. Initialisation
   Possibly using a Hadamard gate on an initial state like $|0\rangle$. The result is distribution over the all states, all with equal probability.
   $$|I_0\rangle = [\underbrace{\frac{1}{\sqrt{N}}, ..., \frac{1}{\sqrt{N}}}_{N \text{ times}}]$$
   It is clear that from the amplitudes, all probabilities are $\frac{1}{N}$ and thus uniform. All states are equally likely to be observed.

2. Oracle gate
   We apply an operation, using the oracle, on our system that flips the sign of our sought after item $\bar{x}$. The operation is defined as follows:
   $$U_f = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & -1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \qquad \text{like identity matrix } I_N \text{ with } (U_f)_{kk} = -1 \text{ for } f(x_k) = 1$$
   This operation is equivalent with a rotation of π radians on the phase of the wanted item, because $e^{i\pi} = -1$. This corresponds with the first plot of figure 1.

3. Grover Operation
   This operation is what does the inversion about average. In other words, it flips all amplitudes over the average value, while keeping the same average value. If μ is the average, then for some value $x = \mu - \epsilon$ you would get $x^* = \mu + \epsilon$. This is done by the following matrix:
   $$D = \begin{bmatrix} -1+\frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & -1+\frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} & -1+\frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \cdots & -1+\frac{2}{N} \end{bmatrix}. \qquad \text{in general: } D_{ij} = \begin{cases} \frac{2}{N} & \text{if } i \neq j, \\ -1+\frac{2}{N} & \text{if } i = j. \end{cases}$$
   . One can check that than $D = HRH$, for H the Hadamard matrix and R a rotation matrix:
   $$R = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix} \qquad \text{in general: } R_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j \text{ and } i = 0, \\ -1 & \text{if } i = j \text{ and } i \neq 0. \end{cases}$$

. The proof of $D = HRH$ is provided in paper [10], where it just works out the general definition of the matrices. The matrix D can also be described as the inversion about the average; $D = -I_N + 2P$, where P is a projection matrix with some nice properties. The latter has all elements $\frac{1}{N}$, $(P)_{ij} = \frac{1}{N}$, $\forall i, j$, which implies that multiplication with a vector v results in a vector containing everywhere the average of elements in v, $Pv = \left[\sum_{i=1}^{N} \frac{1}{N} \cdot v_i\,, ...\right]$.
The square of P, P^2 is again P, this results in $D^2 = (-I + 2P)^2 = I - 4P + 4P^2 = I_N$. Because the conjugate transpose of P is again itself, it is a $N \times N$ matrix with all elements the same non-complex value, the matrix D is also unitary [5] and thus well-defined and keeps norm one by law of total probability.
To show the inversion about the average, take a random vector v. We will notate the average vector as $\overrightarrow{\mu} := Pv$, with Pv's elements all being the scalar average μ. For each element in v, we can write it as a difference with the average μ; $\forall v_i \in v\,, \exists \epsilon_i > 0 :\ v_i = \mu \pm \epsilon_i$. We write ϵ as the vector containing all those values ϵ_i. We have for multiplication with $D = -I_N + 2P$:

$$Dv = -v + 2\overrightarrow{\mu} = (-\overrightarrow{\mu} \mp \epsilon) + 2\overrightarrow{\mu} = \overrightarrow{\mu} \mp \epsilon \tag{1}$$

This clearly flips over the average. To see that the average remains the same, we can write it as follows:

$$\begin{aligned} \mu =& \frac{1}{N}\sum_{i=1}^{N} v_i = \frac{1}{N}\sum_{i=1}^{N}(\mu \pm \epsilon_i) = \mu + \frac{1}{N}\sum_{i=1}^{N}\epsilon_i \\ \implies & \frac{1}{N}\sum_{i=1}^{N}\epsilon_i = 0 \end{aligned}$$

Changing the sign for the epsilon values thus does not change the value of μ. This operation is visible in figure 1. Steps 2-3 are repeated, and then the system is measured.

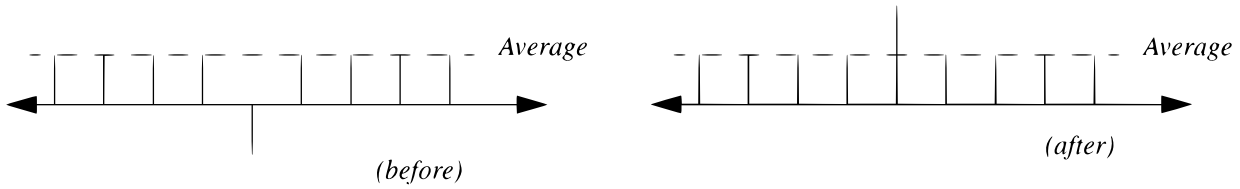

Fig. 1: Images from the original paper [10]; Display of amplitudes, linked with probability. Before plot is after sign flip, after displays the inversion about average, amplifying a given state

3.1 Time Complexity

We now see how this Grover operation amplifies a state from initialisation. This process, step 2 to 3, can be repeated to make the amplitude even higher. This value keeps getting bigger, because when the amplitude is made negative, notate $-k_1$, its distance with the average μ is $\mu + k_1$. When flipping it, it thus becomes that distance on the other side of the average, $k_2 = \mu + k_1 + \mu$. When repeated, this will thus grow bigger and bigger. Here, the lower index indicates iterations. From k_1 to

$$k_2 = -(-k_1) + 2\mu > k_1 \tag{2}$$

. That is, if the average stays positive. If we notate the amplitude of our wanted item $\bar{x}$ as k and the amplitude of the other $N-1$ items as l, the average is $\mu(k,l) := l + \frac{k-l}{N}$. And so $\mu(k,l) > 0 \Leftrightarrow l(N-1)+k > 0$. This is in order in the beginning, because $l > 0$ and $k > 0$, assuming we have a database with bigger size than one. After that, the values for k and l keep changing. More specific:

$$k \xrightarrow{\text{sign flip}} -k \xrightarrow{\text{inv. over avg.}} -(-k) + 2\mu(-k,l) \qquad l \xrightarrow{\text{inv. over avg.}} -l + 2\mu(-k,l)$$

Knowing this and μ in terms of k and l, assuming $N > 2$ non-trivial, we can make the following observation for iterations:

$$\begin{bmatrix} k_i \\ l_i \end{bmatrix} = \frac{1}{N} \begin{bmatrix} N-2 & 2N-2 \\ -2 & N-2 \end{bmatrix} \begin{bmatrix} k_{i-1} \\ l_{i-1} \end{bmatrix} \tag{3}$$

As mentioned in section 1, we want a probability of $\frac{1}{2}$, and thus an amplitude of $k_i = \frac{1}{\sqrt{2}}$. Hence, we will only focus on the k part here

$$k_i = \frac{1}{N}\left[(N-2)k_{i-1} + (2N-2)l_{i-1}\right]$$
$$\Rightarrow k_i - k_{i-1} = \Delta k = \frac{1}{N}(-2k_{i-1} + (2N-2)l_{i-1})$$

Knowing the sum of probabilities must stay one, we can get an upper bound for the value of l_{i-1}.

$$(N-1)l_{i-1}^2 + k_{i-1}^2 = 1 \Longleftrightarrow l_{i-1} = \sqrt{\frac{1-k_{i-1}^2}{N-1}} > \frac{1}{\sqrt{2N-2}}$$

This last bound holds, assuming $0 < k < \frac{1}{\sqrt{2}}$, otherwise we can stop iterating because we achieved the threshold.

$$\Delta k = \frac{1}{N}(-2k_{i-1} + (2N-2)l_{i-1}) > \frac{1}{N}(-2\frac{1}{\sqrt{2}} + \sqrt{2N-2})$$
$$\Delta k > \frac{-\sqrt{2} + \sqrt{2N-2}}{N} > \frac{1}{3\sqrt{N}}$$

This last bound holds for $N > 2$. And thus, in a $\mathcal{O}(\frac{3}{\sqrt{2}}\sqrt{N})$ steps, we will have $k > \frac{1}{\sqrt{2}}$. In the original paper[10], Grover takes a few more stops and shows that each iteration the value of k will grow with $\Delta k > \frac{1}{2\sqrt{N}}$ and thus comes to the conclusion of $\mathcal{O}(\sqrt{N})$ steps.
There are variants of the algorithm with more than one matching solution. These will have different complexity proof depending on when you know the number of possible solutions or not. When the number of solutions is known, complexity reduces to is $\mathcal{O}(\sqrt{N/t})$ [7][3]. If unknown, there is no real a priori knowledge on the amount of iterations but is expected to be also $\mathcal{O}(\sqrt{N/t})$ [13].

Note on the performance of Grover in reality: Grover's original paper does not mention the implementation of the oracle or linking the data to a qubit system. This might be significant. Also, Grover is very resource intensive, making Grover's improvement on attacks not that significant [12][4].

3.2 Lower Bound

As stated in [10], no unstructered search algorithm surpasses $\mathcal{O}(\sqrt{N})$. This because T steps can only consider T^2 queries of the data [1]. If it takes less time than $\mathcal{O}(\sqrt{N})$, it would not take into account all the data, and thus the oracle output could then be altered without affecting the output. This is not possible.
We can conclude that exponentially large search problems still remain exponential with Grover's algorithm, but will be significantly smaller for bigger values of N. For NP problems, it will be significantly faster compared to classical methods [15], but it does not achieve polynomial-time efficiency, unlike Shor's algorithm [5].

4 Cryptographic Implications

4.1 Attacking AES

With Grover in our toolbox, we can take a try at improving a brute force attack on AES. With a given set of random text pair $\{(P, C) \,|\, Enc(k^*, P) = C\}$, plaintext and ciphertext, we can define a function f that will be the oracle for the algorithm. We take r plain- and ciphertext couples to avoid getting *spurious keys*, wrong keys but with same encryption for some plaintext. As described in [12][4], we want for 2 different keys, e.g. the key we want k^* and another, that the encryption is different with probability $1 - \frac{1}{2^{rn}}$. Resulting in

$$r > \lceil 2k/n \rceil \qquad \text{for } k = \text{keylength}, \, n = \text{block size}$$

For AES-128 we take 3 pairs thus. The oracle will take as input a key in the key space, $k \in \mathcal{K}$.

$$f : \mathcal{K} \longrightarrow \{0,1\} : \; k \longmapsto \begin{cases} 1 & E(k, P_1) = C_1 \; \wedge \; E(k, P_2) = C_2 \; \wedge \; E(k, P_3) = C_3 \\ 0 & \text{otherwise} \end{cases}$$

Using AES-128, with a 128-bit key, there are $N = 2^{128}$ different keys. Using Grover, we can find the right key in $\mathcal{O}(\sqrt{N}) \approx 2^{64}$ iterations. As stated in [6], Grover's algorithm is, while optimal, still not very powerful and not quite parallelizable. So by [12][4], AES is still relatively safe, but upgrading the 128-bit keys is encouraged.

4.2 Hash Functions

As suggested in section 1, we can apply Grover to find the preimages of hash functions. One can implement it straightforward with an oracle $f_1 : x \longmapsto \begin{cases} 1 & H(x) = x^* \\ 0 & \text{otherwise} \end{cases}$, where you want a preimage of hash x^*. For a n-bit hash function, this will take $\mathcal{O}(\sqrt{N}) = \mathcal{O}\sqrt{2^n}$ hashes. For finding collisions, you could use the BHT algorithm (Brassard-Høyer-Tapp)[8], which uses Grover's algorithm. Here you calculate and store a table of $\mathcal{O}(\sqrt{N^{1/3}})$ random hashes and their preimages, (r_i, h_i). Then, apply Grover with multiple matching solutions on oracle $f_2 : x \longmapsto \begin{cases} 1 & H(x) = h_i, \, x \neq r_i \\ 0 & \text{otherwise} \end{cases}$. This results in a query complexity of $\mathcal{O}(\sqrt{\frac{N}{N^{1/3}}}) = \mathcal{O}(N^{1/3})$. This all could be used for password cracking [3]. Password salting will prevent the attacker from attacking multiple accounts at once, but won't make it more difficult on one account.

4.3 Prevention

As demonstrated, Grover can do some damage on computing time in theory. But this can be prevented by just doubling the security parameter[11]. Take for example AES-256, using Grover, it would be equivalent to cracking AES-128 by brute force, which will take ages. Grover is nice on paper, but when looking at more literature, its effect is not that impactful.

References

1. G. B. U. V. Charles H. Bennett, Ethan Bernstein. *Strengths and Weaknesses of Quantum Computing.* SIAM Journal on Computing, 1996.
2. M. A. N. . I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2010.
3. H. Corrigan-Gibbs. *Grover Search and Its Cryptographic Applications.* Massachusetts Institute of Technology, 2016.

4. S. D. and P. C. *On the practical cost of Grover for AES key recovery*. UK National Cyber Security Centre, 2024.
5. R. de Wolf. *Quantum Computing: Lecture Notes*. QuSoft, CWI and University of Amsterdam, 2023.
6. S. Fluhrer. *Reassessing Grover's Algorithm*. Cisco Systems, USA, 2017.
7. D. Franklin. *Grover's Algorithm: Searching for quantum advantage with practical applications*. Department of Computer Science at the University of Chicago, 2024.
8. A. T. Gilles Brassard, Peter Høyer. *Quantum Algorithm for the Collision Problem*. 1997.
9. I. Glendinning. *The Bloch Sphere*. European Centre For Parallel Computing At Vienna, 2005.
10. L. K. Grover. *A fast quantum mechanical algorithm for database search*. 1996.
11. A. Hülsing. *Post-quantum cryptography - course Applied Cryptography*. Technische Universiteit Eindhoven TU/e, 2023.
12. M. R. R. S. Markus Grassl, Brandon Langenberg. *Applying Grover's algorithm to AES: quantum resource estimates*. 2015.
13. P. H. A. T. Michel Boyer, Gilles Brassard. *Tight bounds on quantum searching*. PhysComp96, 1996.
14. A. Sudbery. *The life and entangled adventures of Schrödinger's cat*. Department of Mathematics, University of York, Heslington, York, England YO10 5DD, 2023.
15. T. G. Wong. *Introduction to Classical and Quantum Computing*. Rooted Grove, Omaha, Nebraska, 2022.