Losing your Groverhead: new upper bounds and Q# implementations of AES and LowMC

Samuel Jaques[1], Michael Naehrig[2], Martin Roetteler[3], **Fernando Virdia**[4]

[1]Department of Materials, University of Oxford, UK
[2]Microsoft Research, Redmond, WA, USA
[3]Microsoft Quantum, Redmond, WA, USA
[4]Information Security Group, Royal Holloway, University of London, UK

Quantum Cryptanalysis 2019
Schloss Dagstuhl

Overview

- AES
- Grover's search
- Quantum circuits
- Previous work
- Our improvements
- LowMC
- Future directions

- AES [DR01] is a block cipher standardized by NIST in '01.
- Key lengths in $\{128, 192, 256\}$ bits, block size 128 bits.

- AES [DR01] is a block cipher standardized by NIST in '01.
- Key lengths in $\{128, 192, 256\}$ bits, block size 128 bits.
- AES-128 (resp. -192, -256) uses 10 (resp. 12, 14) rounds.

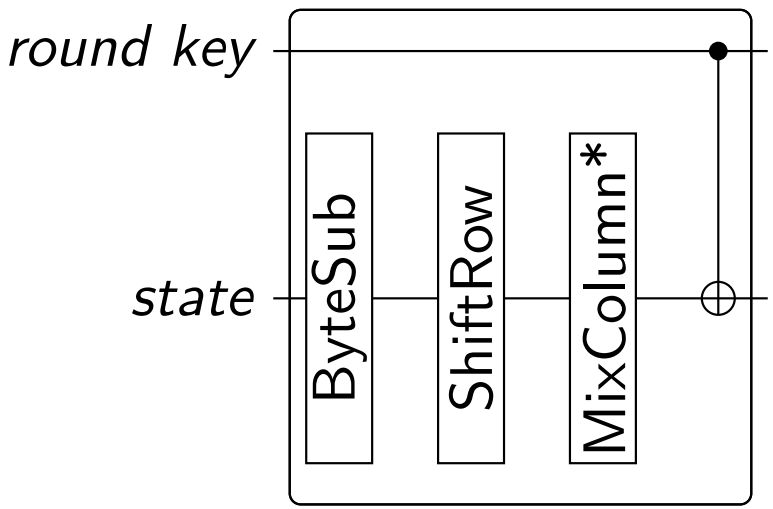


Figure: AES round design.

- The round key is expanded from the ciphers' key, and 128 bits are XOR'd on the state each round.

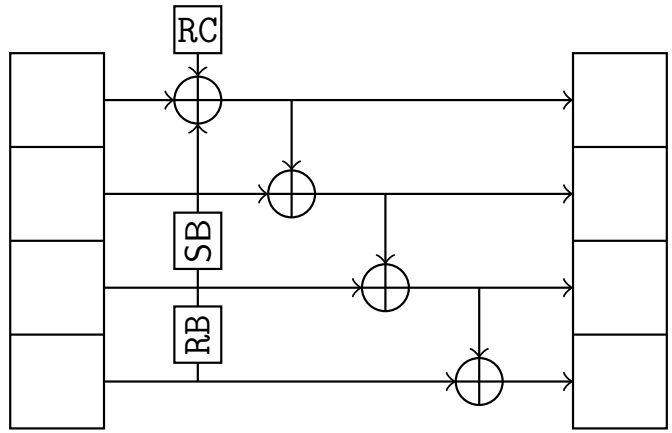


Figure: AES-128 key expansion 'round'. Credit: Jérémy Jean.

- The round key is expanded from the ciphers' key, and 128 bits are XOR'd on the state each round.

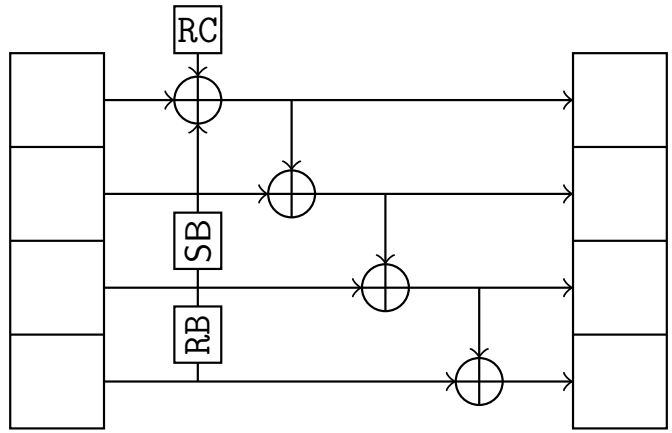


Figure: AES-128 key expansion 'round'. Credit: Jérémy Jean.

- ShiftRow and RotByte are permutations.
- MixColumn is an invertible linear transformation.
- ByteSub and SubByte are byte-wise applications of the S-box, which computes inversion in $\mathbb{F}_{2^8}$ (and maps $0 \mapsto 0$).

Grover's search [Gro96] is a quantum algorithm for finding elements in unsorted lists of size N.

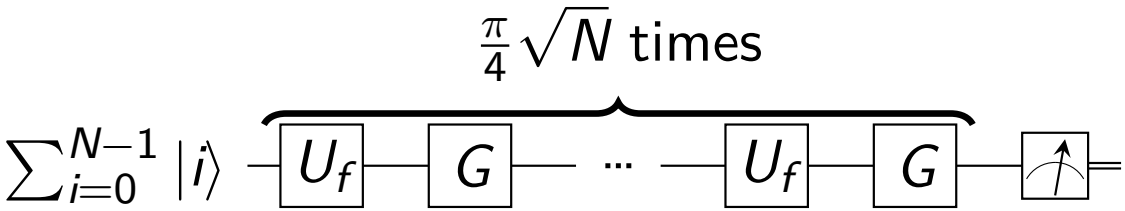


Figure: Grover's search sketch.

- Grover's search [Gro96] is a quantum algorithm for finding elements in unsorted lists of size N.

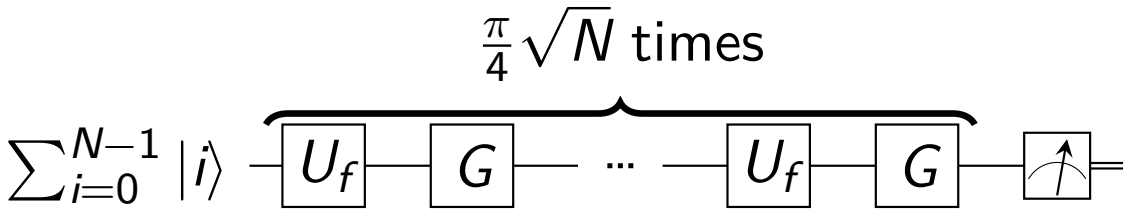


Figure: Grover's search sketch.

- U_f is a quantum circuit mapping $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$, where $f(x) = (x == target)$.
- G is a "reflection around the mean" operation.

- A quantum circuit is a sequence of unitary operators (and measurements).
- Width: the maximum number of qubits used.
- Depth: the number of sequential "basic" operations.
- Complex operations can be constructed from simple (universal) sets of gates. We use Clifford + T.

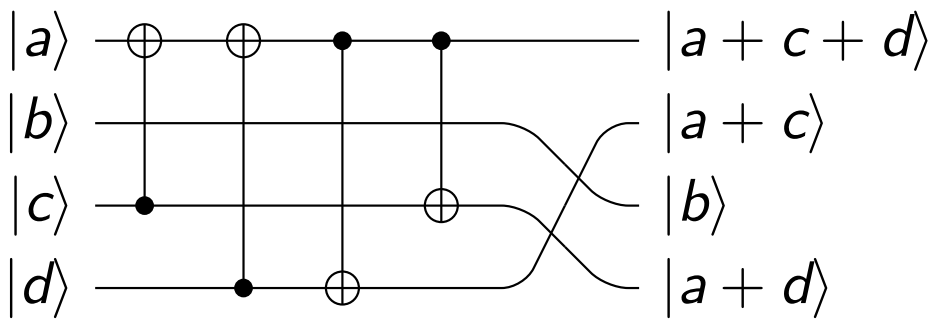


Figure: Quantum circuit example.

In this work we make the following assumptions:

In this work we make the following assumptions:

- We only work with logical qubits.

In this work we make the following assumptions:

- We only work with logical qubits.
- We do not assume any particular framework (e.g. the surface code).
  - Hence no costs for idle qubits or need for gates to operate locally.
  - But also no speedups like free CNOT fan-outs.

In this work we make the following assumptions:

- We only work with logical qubits.
- We do not assume any particular framework (e.g. the surface code).
  - Hence no costs for idle qubits or need for gates to operate locally.
  - But also no speedups like free CNOT fan-outs.
- Swapping qubits is free, by "rewiring" (keeping track of the swaps).

In this work we make the following assumptions:

- We only work with logical qubits.
- We do not assume any particular framework (e.g. the surface code).
  - Hence no costs for idle qubits or need for gates to operate locally.
  - But also no speedups like free CNOT fan-outs.
- Swapping qubits is free, by "rewiring" (keeping track of the swaps).

This is not necessarily "realistic", but is what the previous literature on AES (and hence NIST in [Nat16]) uses.

Now a quick look at basic tools: gates, linear programs, constant matrix multiplication.

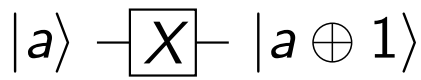
Now a quick look at basic tools: gates, linear programs, constant matrix multiplication.

- X gate, works as an in-place classical NOT.

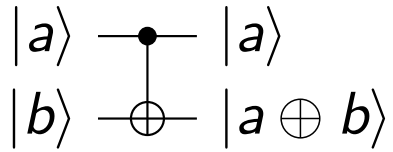
$$|a\rangle \; -\boxed{X}- \; |a \oplus 1\rangle$$

Now a quick look at basic tools: gates, linear programs, constant matrix multiplication.

- X gate, works as an in-place classical NOT.

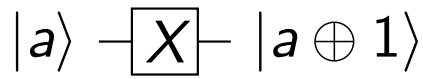


- CNOT gate, works similarly to an XOR gate .

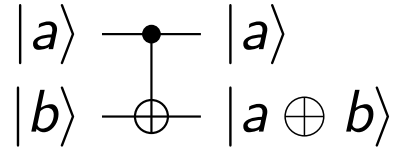


Now a quick look at basic tools: gates, linear programs, constant matrix multiplication.

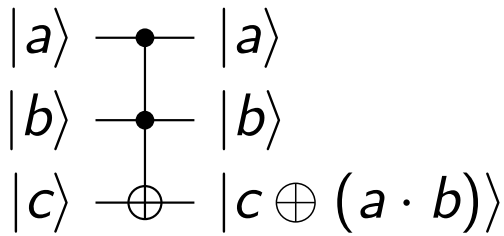
- X gate, works as an in-place classical NOT.



- CNOT gate, works similarly to an XOR gate .



- Toffoli (aka CCNOT), works similarly to an AND gate.



Linear programs are sequences of $w \leftarrow x \star y$, for some binary operation $\star$.

t_1	$=$	$x_2 + x_3$	t_2	$=$	$x_2 \times x_0$	t_3	$=$	$x_1 + t_2$
t_4	$=$	$x_0 + x_1$	t_5	$=$	$x_3 + t_2$	t_6	$=$	$t_5 \times t_4$
t_7	$=$	$t_3 \times t_1$	t_8	$=$	$x_0 \times x_3$	t_9	$=$	$t_4 \times t_8$
t_{10}	$=$	$t_4 + t_9$	t_{11}	$=$	$x_1 \times x_2$	t_{12}	$=$	$t_1 \times t_{11}$
t_{13}	$=$	$t_1 + t_{12}$	y_0	$=$	$t_2 + t_{13}$	y_1	$=$	$x_3 + t_7$
y_2	$=$	$t_2 + t_{10}$	y_3	$=$	$x_1 + t_6$			

Figure 1: Inversion in $GF(2^4)$. Input is (x_0, x_1, x_2, x_3) and output is (y_0, y_1, y_2, y_3).

These can be easily translated into Q# by writing the appropriate self-inverse operators $(x, y, z) \mapsto (x, y, z \oplus (x \star y))$.

Image taken from [BP11].

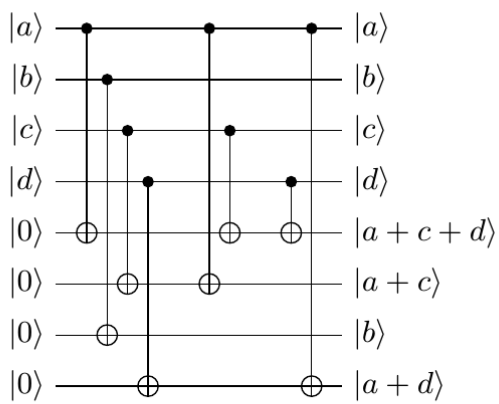
Multiplication by constant matrices:

- Naively, using ancilla qubits (b).
- Rearranging operations from the naive version **will help**.
- Invertible matrices can be inplemented in-place by PLU decomposing them [TB97] (c).

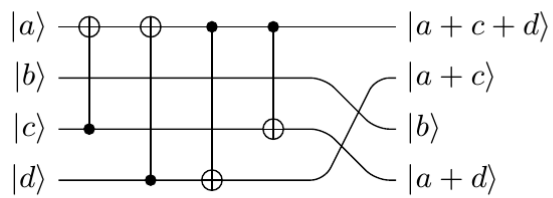
And the P is for free!

$$M = \begin{pmatrix} 1&0&1&1 \\ 1&0&1&0 \\ 0&1&0&0 \\ 1&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 1&0&1&0 \\ 1&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} 1&0&1&1 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} = P \cdot L \cdot U$$

(a) Invertible linear transformation M and its PLU decomposition.



(b) Naive circuit computing M.



(c) In-place implementation of M.

Fig. 1. Alternative circuits implementing the same linear transformation $M : \mathbb{F}_2^4 \to \mathbb{F}_2^4$, by using the two strategies described in Section 2.3.

- [YI00] introduces the idea of using Grover's to attack block ciphers with keylength k in $O(2^{k/2})$ "operations" (U_f and G).
- Grassl et al. [GLRS16] provide the first cost estimate for U_f (and disregard G, we do the same!).

Component	[GLRS16] design
ShiftRow, RotByte	Rewiring (free)
MixColumn	Multiplication by constant in $\mathbb{F}_{2^8}[x]/(x^4+1)$: Invertible linear map, hence PLU decomposition
S-box	Inversion $\equiv ((\alpha \cdot \alpha^2) \cdot (\alpha \cdot \alpha^2)^4 \cdot (\alpha \cdot \alpha^2)^{16} \cdot \alpha^{64})^2$ in $\mathbb{F}_2[x]/\langle x^8 + x^4 + x^3 + x + 1\rangle$:
AddRoundKey	Bitwise XOR/CNOT
KeyExpansion	Caching of "expensive" bytes + recomputing of "cheap" ones
Rounds	"Pebbling" to reduce the number of ancilla qubits

When computing rounds, they try to tradeoff between circuit width and depth, using a "pebbling" strategy.

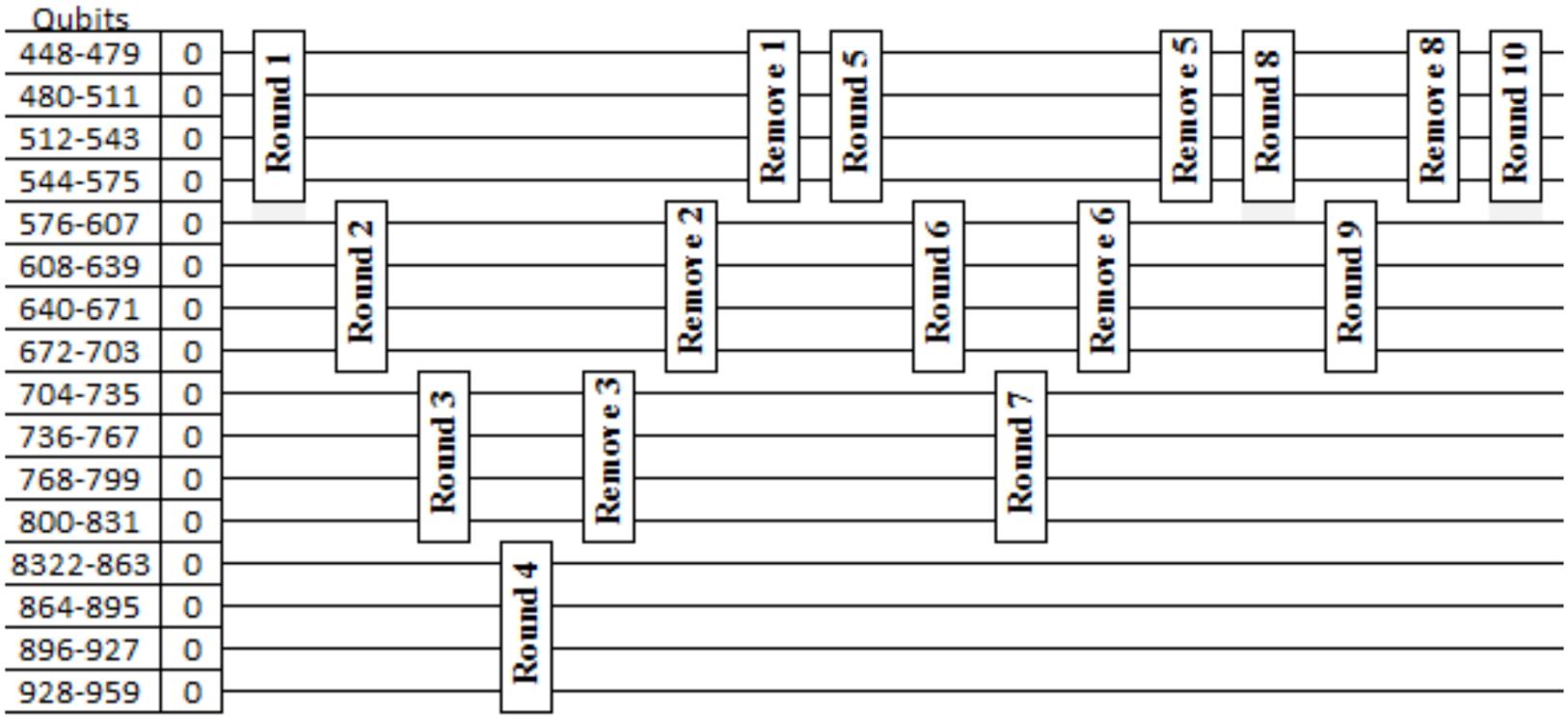


Figure: Round pebbling strategy for AES-128 in [GLRS16].

- Given any block cipher C: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, for any (m, c) pair there may be multiple K such that $C(K, m) = c$.
- Hence, when doing exhaustive key search, multiple (m_i, c_i) pairs may be needed to uniquely determine a key K whp.
- [GLRS16] find that for AES-128 (resp. -192, -256), 3 (resp. 4, 5) pairs are needed when implementing U_f.

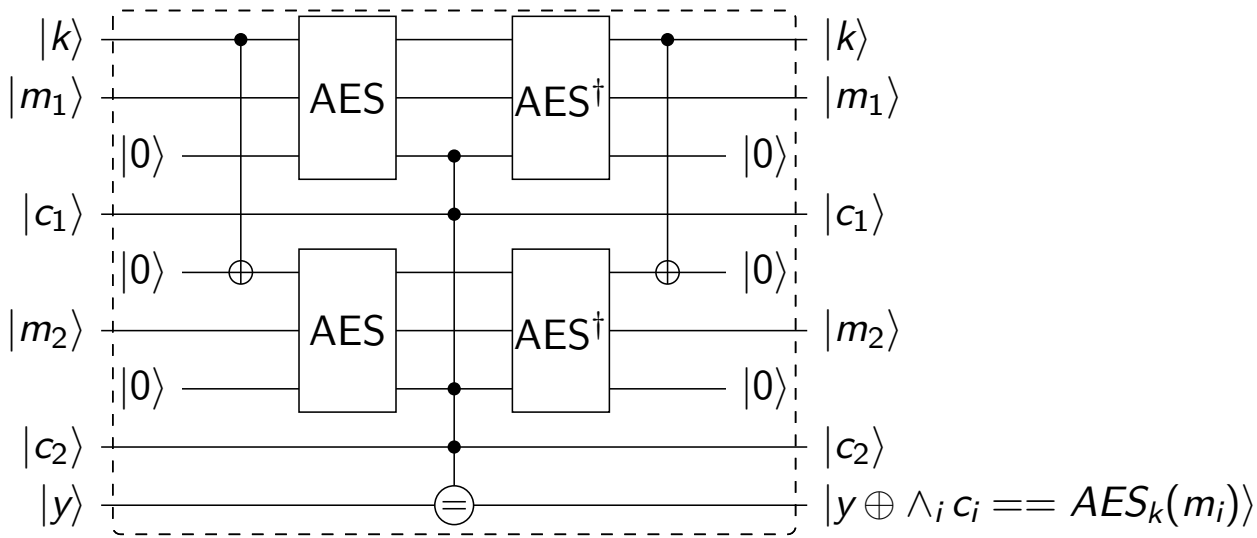


Figure: Example AES U_f for two (m_i, c_i) pairs.

Overall, Grassl et al. estimate the cost of U_f as follows.

scheme	#(1qCliff+CNOT)	#T	T-depth	full depth	width	G-cost	DW-cost
AES-128	$1.55 \cdot 2^{86}$	$1.19 \cdot 2^{86}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2 953	$1.37 \cdot 2^{87}$	$1.67 \cdot 2^{92}$
AES-192	$1.17 \cdot 2^{119}$	$1.81 \cdot 2^{118}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4 449	$1.04 \cdot 2^{120}$	$1.44 \cdot 2^{125}$
AES-256	$1.83 \cdot 2^{151}$	$1.41 \cdot 2^{151}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6 681	$1.62 \cdot 2^{152}$	$1.28 \cdot 2^{158}$

Table: Circuit size for U_f as in [GLRS16].

- In 2016, NIST puts out a call for post-quantum cryptography proposals [Nat16].
- Regarding quantum computation capabilities, they suggest having a $\texttt{MAXDEPTH} \in \{2^{40}, 2^{64}, 2^{96}\}$ parameter bounding quantum computation depth.
- They also define security "categories" 1, 3, and 5, based on the hardness of key recovery against AES-128, -192, -256.

- In 2016, NIST puts out a call for post-quantum cryptography proposals [Nat16].
- Regarding quantum computation capabilities, they suggest having a $\texttt{MAXDEPTH} \in \{2^{40}, 2^{64}, 2^{96}\}$ parameter bounding quantum computation depth.
- They also define security "categories" 1, 3, and 5, based on the hardness of key recovery against AES-128, -192, -256.
- Early termination of Grover's search results in low success probabilities.
- Hence, due to `MAXDEPTH`, Grover's search against AES needs to be parallelised.

For Grover's search, Zalka [Zal99] showed that using S machines saves only $\sqrt{S}$ depth, optimally.

- For Grover's search, Zalka [Zal99] showed that using S machines saves only $\sqrt{S}$ depth, optimally.

- NIST's reasoning:
  - Say non-parallel Grover's search requires depth $D = x \cdot \texttt{MAXDEPTH}$, for some $x \geq 1$ and G gates.

- For Grover's search, Zalka [Zal99] showed that using S machines saves only $\sqrt{S}$ depth, optimally.

- NIST's reasoning:
  - Say non-parallel Grover's search requires depth $D = x \cdot \texttt{MAXDEPTH}$, for some $x \geq 1$ and G gates.
  - To cut depth by x, x^2 machines are needed. Each uses $\approx G/x$ gates.

- For Grover's search, Zalka [Zal99] showed that using S machines saves only $\sqrt{S}$ depth, optimally.

- NIST's reasoning:
  - Say non-parallel Grover's search requires depth $D = x \cdot \texttt{MAXDEPTH}$, for some $x \geq 1$ and G gates.
  - To cut depth by x, x^2 machines are needed. Each uses $\approx G/x$ gates.
  - Total gate count: $(G/x) \cdot x^2 = G \cdot D/\texttt{MAXDEPTH}$.

- For Grover's search, Zalka [Zal99] showed that using S machines saves only $\sqrt{S}$ depth, optimally.

- NIST's reasoning:
  - Say non-parallel Grover's search requires depth $D = x \cdot \texttt{MAXDEPTH}$, for some $x \geq 1$ and G gates.
  - To cut depth by x, x^2 machines are needed. Each uses $\approx G/x$ gates.
  - Total gate count: $(G/x) \cdot x^2 = G \cdot D/\texttt{MAXDEPTH}$.

- Using D and G from [GLRS16], they deduce the security categories' requirements.

AES 128	2^{170}/MAXDEPTH quantum gates
AES 192	2^{233}/MAXDEPTH quantum gates
AES 256	2^{298}/MAXDEPTH quantum gates

Figure: Definition for security categories 1, 3, and 5.

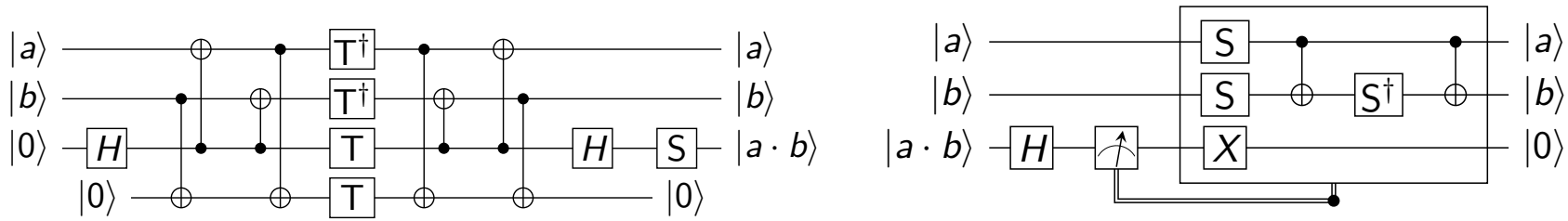
- Our initial idea: NIST cares about limiting depth, but uses [GLRS16] which optimizes for width. What if we minimize depth?
- Hindsight: parallelisation is bad, so crucially beneficial to minimise depth!
- We also get a Q# implementation:
  - testable,
  - friendly to read/modify,
  - automated circuit size estimates,
  - easy to translate linear programs/verilog using regexes!

We look now at our design choices for a smaller Grover oracle for AES.

S-box: well investigated in the hardware literature.

- Lots of linear programs to translate and test.
- Tried various variants of [BP11].
- Scooped! In concurrent indepedent work, Langenberg et al. [LPS19] propose a similar S-box change.
  - They keep the same pebbling strategy of [GLRS16] and provide only an implementation of their S-box.

- Inside their S-box, [GLRS16] use a 7 T-gates implementation of Toffoli.
- We replace Toffoli with AND gates, using a custom design by Mathias Soeken, based on Selinger [Sel13] and Gidney [Gid18].



(a) AND gate.

(b) AND† gate.

Figure: AND gate with T-depth 1, T count 4, and "T-free" adjoint operator. It does introduce measurements.

- KeyExpansion: instead of caching, we do in-place expansion as necessary.

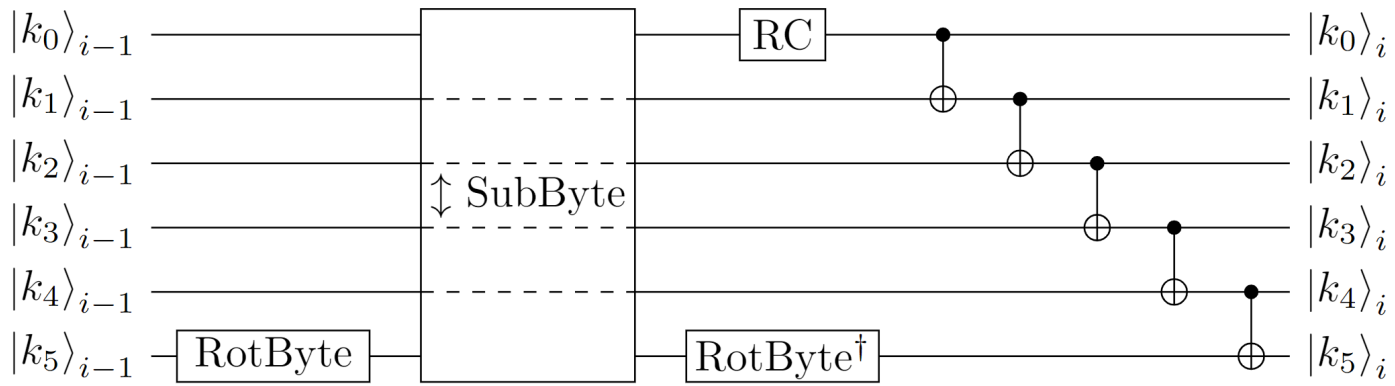


Figure: AES 192 in-place i^{th} round key expansion.

- This saves us qubits with respect to naive full expansion, while not increasing depth due to the computations running in parallel to the round.

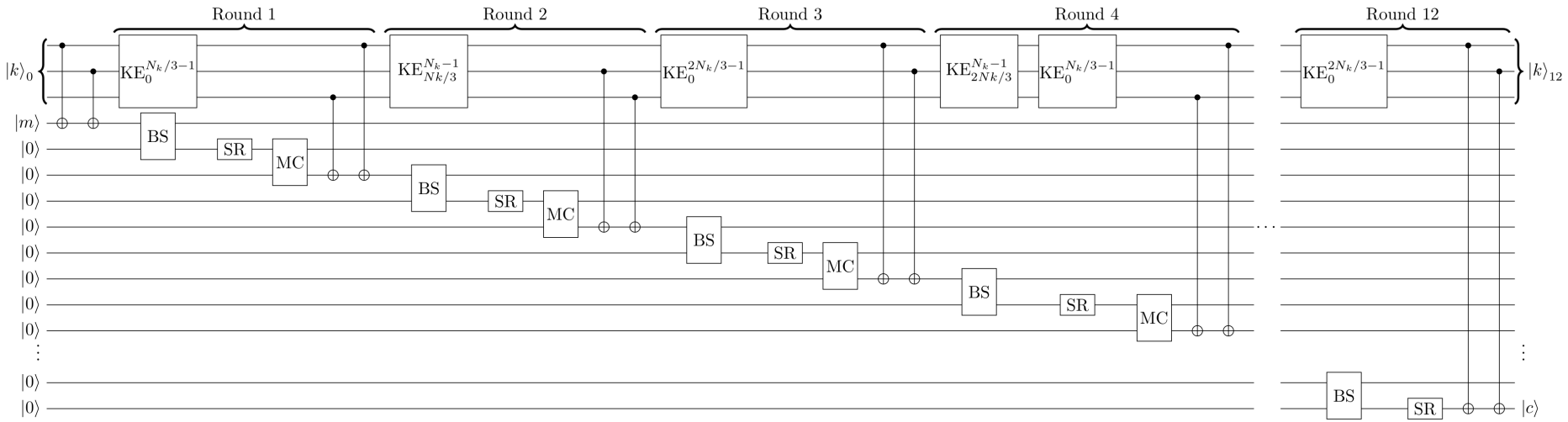


Figure: AES 192 round structure.

Indeed, rounds only require 128 bits of expanded key at every time.

Other improvements:

- We cost both the PLU-decomposed in-place MixColumn design, and a recent, shallower (but wider) design by Maximov [Max19].
- Fix to the key uniqueness computation: 3, 4, 5 pairs are too many!
  - For $p \approx 1$ attacks, 2, 2, 3 pairs are enough.
  - As Langenberg et al. [LPS19] also noticed, we suggest using 1, 2, 2 pairs for high probability attacks ($\approx 1/e$, ≈ 1, $\approx 1/e$) when using unbounded Grover's.

scheme	#(1qCliff+CNOT)	#T	#M	T-depth	full depth	width	*G*-cost	*DW*-cost	p_{succ}
Grassl et al. [GLRS16]									
AES-128 ($r=3$)	$1.55 \cdot 2^{86}$	$1.19 \cdot 2^{86}$	0	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2 953	$1.37 \cdot 2^{87}$	$1.67 \cdot 2^{92}$	≈ 1
AES-192 ($r=4$)	$1.17 \cdot 2^{119}$	$1.81 \cdot 2^{118}$	0	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4 449	$1.04 \cdot 2^{120}$	$1.44 \cdot 2^{125}$	≈ 1
AES-256 ($r=5$)	$1.83 \cdot 2^{151}$	$1.41 \cdot 2^{151}$	0	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6 681	$1.62 \cdot 2^{152}$	$1.28 \cdot 2^{158}$	≈ 1
Langenberg et al. [LPS19]									
AES-128 ($r=1$)	$1.46 \cdot 2^{82}$	$1.47 \cdot 2^{81}$	0	$1.44 \cdot 2^{77}$	$1.39 \cdot 2^{79}$	865	$1.10 \cdot 2^{83}$	$1.17 \cdot 2^{89}$	$\approx 1/e$
AES-192 ($r=2$)	$1.71 \cdot 2^{115}$	$1.68 \cdot 2^{114}$	0	$1.26 \cdot 2^{109}$	$1.23 \cdot 2^{111}$	1 793	$1.27 \cdot 2^{116}$	$1.08 \cdot 2^{122}$	≈ 1
AES-256 ($r=2$)	$1.03 \cdot 2^{148}$	$1.02 \cdot 2^{147}$	0	$1.66 \cdot 2^{141}$	$1.61 \cdot 2^{143}$	2 465	$1.54 \cdot 2^{148}$	$1.94 \cdot 2^{154}$	$\approx 1/e$
this work									
AES-128 (IP MC, $r=1$)	$1.13 \cdot 2^{82}$	$1.32 \cdot 2^{79}$	$1.32 \cdot 2^{77}$	$1.48 \cdot 2^{70}$	$1.08 \cdot 2^{75}$	1665	$1.33 \cdot 2^{82}$	$1.76 \cdot 2^{85}$	$\approx 1/e$
AES-128 (IP MC, $r=2$)	$1.13 \cdot 2^{83}$	$1.32 \cdot 2^{80}$	$1.32 \cdot 2^{78}$	$1.48 \cdot 2^{70}$	$1.08 \cdot 2^{75}$	3329	$1.34 \cdot 2^{83}$	$1.75 \cdot 2^{86}$	≈ 1
AES-192 (IP MC, $r=2$)	$1.27 \cdot 2^{115}$	$1.47 \cdot 2^{112}$	$1.47 \cdot 2^{110}$	$1.47 \cdot 2^{102}$	$1.14 \cdot 2^{107}$	3969	$1.50 \cdot 2^{115}$	$1.11 \cdot 2^{119}$	≈ 1
AES-256 (IP MC, $r=2$)	$1.56 \cdot 2^{147}$	$1.81 \cdot 2^{144}$	$1.81 \cdot 2^{142}$	$1.55 \cdot 2^{134}$	$1.29 \cdot 2^{139}$	4609	$1.84 \cdot 2^{147}$	$1.45 \cdot 2^{151}$	$\approx 1/e$
AES-256 (IP MC, $r=3$)	$1.17 \cdot 2^{148}$	$1.36 \cdot 2^{145}$	$1.36 \cdot 2^{143}$	$1.55 \cdot 2^{134}$	$1.28 \cdot 2^{139}$	6913	$1.38 \cdot 2^{148}$	$1.08 \cdot 2^{152}$	≈ 1

Table: Comparison of cost estimates for Grover's algorithm with $\left\lfloor \frac{\pi}{4} 2^{k/2} \right\rfloor$ AES oracle iterations, minimising *G*-cost (in-place MixColumn beats Maximov's here).

- What can we do about the depth constraint? AES-128 in $\texttt{MAXDEPTH} = 2^{96}$ is the only attack fitting.
- Boyer et al. [BBHT98] propose the following when searching a list of size N:
  1. Run $0.583\sqrt{N}$ Grover iterations.
  2. Measure. If output is wrong, go to 1.
- In theory, the expected number of iterations to win becomes $0.690\sqrt{N} < \frac{\pi}{4}\sqrt{N}$.
- In practice, most often one needs to repeat step 1. at least twice $\implies$ $1.166\sqrt{N} > \frac{\pi}{4}\sqrt{N}$ iterations.

Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.

- Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.
- "Outer parallelisation":
  - Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations independently.

- Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.
- "Outer parallelisation":
  - Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations independently.
  - Measure a candidate solution from each machine, and classically check them.

- Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.
- "Outer parallelisation":
  - Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations independently.
  - Measure a candidate solution from each machine, and classically check them.
  - Total success probability is $p_S(j) = 1 - (1 - p(j))^S$, where $p(j)$ is the success probability for a single machine.

- Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.
- "Outer parallelisation":
  - Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations independently.
  - Measure a candidate solution from each machine, and classically check them.
  - Total success probability is $p_S(j) = 1 - (1 - p(j))^S$, where $p(j)$ is the success probability for a single machine.
  - We want to reduce depth by $\sqrt{S}$. Then, S machines $\implies j = \frac{\pi}{4}\sqrt{\frac{N}{S}}$ iterations. As $S \to \infty$, $p_S(j) \to 0.915$.

- Only chance we have is parallelising. Two strategies, using Kim, Han, and Jeong [KHJ18] nomenclature.
- "Outer parallelisation":
  - Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations independently.
  - Measure a candidate solution from each machine, and classically check them.
  - Total success probability is $p_S(j) = 1 - (1 - p(j))^S$, where $p(j)$ is the success probability for a single machine.
  - We want to reduce depth by $\sqrt{S}$. Then, S machines $\implies j = \frac{\pi}{4}\sqrt{\frac{N}{S}}$ iterations. As $S \to \infty$, $p_S(j) \to 0.915$.
  - Hence there is no "outer" strategy with $p \approx 1$ that saves $\sqrt{S}$ depth.

- "Inner parallelisation":
  - The total search space has size N. Partition it into S disjoint subsets. Only one subset contains the correct key.

"Inner parallelisation":

- The total search space has size N. Partition it into S disjoint subsets. Only one subset contains the correct key.
- Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations, each on a different subset of size N/S.

"Inner parallelisation":

- The total search space has size N. Partition it into S disjoint subsets. Only one subset contains the correct key.
- Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations, each on a different subset of size N/S.
- Measure a candidate solution from each machine, and classically check them.

"Inner parallelisation":

- The total search space has size N. Partition it into S disjoint subsets. Only one subset contains the correct key.
- Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations, each on a different subset of size N/S.
- Measure a candidate solution from each machine, and classically check them.
- We want to reduce depth by $\sqrt{S}$. Again, S machines $\implies j = \frac{\pi}{4}\sqrt{\frac{N}{S}}$ iterations. But now, these are the right number of iterations to find the key with $p \approx 1$ in its subset of size N/S!

"Inner parallelisation":

- The total search space has size N. Partition it into S disjoint subsets. Only one subset contains the correct key.
- Have S machines run $j \leq \frac{\pi}{4}\sqrt{N}$ iterations, each on a different subset of size N/S.
- Measure a candidate solution from each machine, and classically check them.
- We want to reduce depth by $\sqrt{S}$. Again, S machines $\implies j = \frac{\pi}{4}\sqrt{\frac{N}{S}}$ iterations. But now, these are the right number of iterations to find the key with $p \approx 1$ in its subset of size N/S!
- In all but one subset we measure a wrong key, in the right subset we measure the correct key. Classically check each, to win with probability $p \approx 1$.

- There's a further advantage of inner parallelisation, when looking for the right key K.
- Take AES-128. We said we need 2 plaintext-ciphertext pairs to uniquely identify $K \in \mathbb{K} = \{0,1\}^{128}$ (i.e. whp no other keys map $m_i \mapsto c_i$ for $i = 1, 2$).

- There's a further advantage of inner parallelisation, when looking for the right key K.
- Take AES-128. We said we need 2 plaintext-ciphertext pairs to uniquely identify $K \in \mathbb{K} = \{0,1\}^{128}$ (i.e. whp no other keys map $m_i \mapsto c_i$ for $i = 1, 2$).
- Using 1 pair, the probability of only K mapping $m \mapsto c$ exists in $\mathbb{K}$ is $1/e$.

- There's a further advantage of inner parallelisation, when looking for the right key K.
- Take AES-128. We said we need 2 plaintext-ciphertext pairs to uniquely identify $K \in \mathbb{K} = \{0,1\}^{128}$ (i.e. whp no other keys map $m_i \mapsto c_i$ for $i = 1, 2$).
- Using 1 pair, the probability of only K mapping $m \mapsto c$ exists in $\mathbb{K}$ is $1/e$.
- Now partition $\mathbb{K}$ into S subsets, say $K \in \mathbb{K}_K \subset \mathbb{K}$. The probability that another "spurious" key mapping $m \mapsto c$ exists in $\mathbb{K}_K$ is now smaller than $1 - 1/e$.
- Hence using inner parallelisation increases the success probability of the attack when using 1 plaintext-ciphertext pair. It works similarly for AES-192 and AES-256.

scheme	MD	r	S	$\log_2(\text{SKP})$	D	W	G-cost	DW-cost
AES-128	2^{40}	1	$1.28 \cdot 2^{69}$	-69.36	$1.00 \cdot 2^{40}$	$1.76 \cdot 2^{80}$	$1.07 \cdot 2^{117}$	$1.76 \cdot 2^{120}$
AES-192	2^{40}	1	$1.04 \cdot 2^{133}$	-69.05	$1.00 \cdot 2^{40}$	$1.72 \cdot 2^{144}$	$1.09 \cdot 2^{181}$	$1.72 \cdot 2^{184}$
AES-256	2^{40}	1	$1.12 \cdot 2^{197}$	-69.16	$1.00 \cdot 2^{40}$	$1.08 \cdot 2^{209}$	$1.39 \cdot 2^{245}$	$1.08 \cdot 2^{249}$
AES-128	2^{64}	1	$1.28 \cdot 2^{21}$	-21.36	$1.00 \cdot 2^{64}$	$1.76 \cdot 2^{32}$	$1.07 \cdot 2^{93}$	$1.76 \cdot 2^{96}$
AES-192	2^{64}	1	$1.04 \cdot 2^{85}$	-21.05	$1.00 \cdot 2^{64}$	$1.72 \cdot 2^{96}$	$1.09 \cdot 2^{157}$	$1.72 \cdot 2^{160}$
AES-256	2^{64}	1	$1.12 \cdot 2^{149}$	-21.16	$1.00 \cdot 2^{64}$	$1.08 \cdot 2^{161}$	$1.39 \cdot 2^{221}$	$1.08 \cdot 2^{225}$
AES-128*	2^{96}	2	$1.00 \cdot 2^{0}$	$-\infty$	$1.08 \cdot 2^{75}$	$1.63 \cdot 2^{11}$	$1.34 \cdot 2^{83}$	$1.75 \cdot 2^{86}$
AES-192	2^{96}	2	$1.05 \cdot 2^{21}$	$-\infty$	$1.00 \cdot 2^{96}$	$1.74 \cdot 2^{33}$	$1.09 \cdot 2^{126}$	$1.74 \cdot 2^{129}$
AES-256	2^{96}	2	$1.12 \cdot 2^{85}$	-85.16	$1.00 \cdot 2^{96}$	$1.09 \cdot 2^{98}$	$1.39 \cdot 2^{190}$	$1.09 \cdot 2^{194}$

Table: Cost estimates for *inner* parallelization. r is the number of plaintext-ciphertext pairs used. SKP is the probability that spurious keys are present in $\mathbb{K}_K$. All circuits use Maximov's [Max19] MixColumns (shallower designs have a better advantage when parallelising) except for AES-128 at $\texttt{MAXDEPTH} = 2^{96}$.

Some observations:

- Say a candidate scheme for category 5 does a similar analysis, and the best quantum attack with $\texttt{MAXDEPTH} = 2^{40}$ requires $S = 2^{230}$ G-cost.

Some observations:

- Say a candidate scheme for category 5 does a similar analysis, and the best quantum attack with $\texttt{MAXDEPTH} = 2^{40}$ requires $S = 2^{230}$ G-cost.
  - Does it not meet the criteria? Nobody is going to build 2^{197} quantum computers anyway, so Grover is not really an attack against AES-256 there.

Some observations:

- Say a candidate scheme for category 5 does a similar analysis, and the best quantum attack with $\texttt{MAXDEPTH} = 2^{40}$ requires $S = 2^{230}$ G-cost.
  - Does it not meet the criteria? Nobody is going to build 2^{197} quantum computers anyway, so Grover is not really an attack against AES-256 there.
- Logical qubits better be free.

Some observations:

- Say a candidate scheme for category 5 does a similar analysis, and the best quantum attack with $\texttt{MAXDEPTH} = 2^{40}$ requires $S = 2^{230}$ G-cost.
  - Does it not meet the criteria? Nobody is going to build 2^{197} quantum computers anyway, so Grover is not really an attack against AES-256 there.
- Logical qubits better be free. Should we introduce `MAXWIDTH`? What would it mean?
  - Maybe that we try to fit Grover within `MAXWIDTH`, compute the success probability for the resulting attack, and then do the same for candidates ("Cat 5, `MD` 2^{40}, `MW` x means no quantum attack with success prob $\geq 2^{-\cdots}$")?

Finally, we can recompute NIST's table, taking into account inner parallelisation advantages.

Finally, we can recompute NIST's table, taking into account inner parallelisation advantages.

NIST Security Strength Category	source	G-cost for MAXDEPTH 2^{40}	2^{64}	2^{96}	approximation
1 AES-128	[Nat16]	2^{130}	2^{106}	2^{74}	$2^{170}/\texttt{MAXDEPTH}$
	this work	$1.07 \cdot 2^{117}$	$1.07 \cdot 2^{93}$	$^{*}1.34 \cdot 2^{83}$	$\approx 2^{157}/\texttt{MAXDEPTH}$
3 AES-192	[Nat16]	2^{193}	2^{169}	2^{137}	$2^{233}/\texttt{MAXDEPTH}$
	this work	$1.09 \cdot 2^{181}$	$1.09 \cdot 2^{157}$	$1.09 \cdot 2^{126}$	$\approx 2^{221}/\texttt{MAXDEPTH}$
5 AES-256	[Nat16]	2^{258}	2^{234}	2^{202}	$2^{298}/\texttt{MAXDEPTH}$
	this work	$1.39 \cdot 2^{245}$	$1.39 \cdot 2^{221}$	$1.39 \cdot 2^{190}$	$\approx 2^{285}/\texttt{MAXDEPTH}$

Table: *approximation* displays the formula used by NIST in [Nat16] for NIST numbers and a rough approximation that would replace the NIST formula based on our results.

LowMC

- LowMC [ARS+15, ARS+16] is a block cipher family designed for FHE and MPC.
- It is used as part of the Picnic [ZCD+17] submission for post-quantum digital signatures.
- We used the same tools used for AES.

scheme	MD	r	S	$\log_2(\text{SKP})$	D	W	G-cost	DW-cost
LowMC L1	2^{40}	1	$1.40 \cdot 2^{80}$	-80.48	$1.00 \cdot 2^{40}$	$1.08 \cdot 2^{91}$	$1.25 \cdot 2^{123}$	$1.08 \cdot 2^{131}$
LowMC L3	2^{40}	1	$1.83 \cdot 2^{147}$	-147.87	$1.00 \cdot 2^{40}$	$1.06 \cdot 2^{159}$	$1.65 \cdot 2^{190}$	$1.06 \cdot 2^{199}$
LowMC L5	2^{40}	1	$1.08 \cdot 2^{214}$	-214.11	$1.00 \cdot 2^{40}$	$1.61 \cdot 2^{225}$	$1.99 \cdot 2^{256}$	$1.61 \cdot 2^{265}$
LowMC L1	2^{64}	1	$1.40 \cdot 2^{32}$	-32.48	$1.00 \cdot 2^{64}$	$1.08 \cdot 2^{43}$	$1.25 \cdot 2^{99}$	$1.08 \cdot 2^{107}$
LowMC L3	2^{64}	1	$1.83 \cdot 2^{99}$	-99.87	$1.00 \cdot 2^{64}$	$1.06 \cdot 2^{111}$	$1.65 \cdot 2^{166}$	$1.06 \cdot 2^{175}$
LowMC L5	2^{64}	1	$1.08 \cdot 2^{166}$	-166.11	$1.00 \cdot 2^{64}$	$1.61 \cdot 2^{177}$	$1.99 \cdot 2^{232}$	$1.61 \cdot 2^{241}$
LowMC L1	2^{96}	2	$1.00 \cdot 2^{0}$	$-\infty$	$1.18 \cdot 2^{80}$	$1.55 \cdot 2^{11}$	$1.06 \cdot 2^{84}$	$1.83 \cdot 2^{91}$
LowMC L3	2^{96}	1	$1.83 \cdot 2^{35}$	-35.87	$1.00 \cdot 2^{96}$	$1.06 \cdot 2^{47}$	$1.65 \cdot 2^{134}$	$1.06 \cdot 2^{143}$
LowMC L5	2^{96}	1	$1.08 \cdot 2^{102}$	-102.11	$1.00 \cdot 2^{96}$	$1.61 \cdot 2^{113}$	$1.99 \cdot 2^{200}$	$1.61 \cdot 2^{209}$

Table: Cost estimates for parallel Grover key search against LowMC under a depth limit MAXDEPTH with *inner* parallelization.

Further research directions:

- Improve the AES oracle with better S-boxes
  - Sacrificing simulatability, it would be possible to use a compiler based on [GKMR14, ZC19] to automatically synthetise smaller circuits.
  - An orthogonal automatic technique could be to use the classical circuit minimizer by [MSR+19, MSC+19] to attempt to further reduce the linear program components.
- Improve the LowMC design by adopting the approach from [DKP+19].
- Redo the analysis in the surface code setting (it would require new implementations probably, maybe a specific surface-code compiler).

- Take some of the quantum algorithms proposed for the candidates (most use Grover), and do a similar analysis of their quantum component. Do they always/never/sometimes hit `MAXDEPTH`?
- Maybe implementing some of these quantum attacks in Q# could give insight.
- What happens if we introduce `MAXWIDTH`? Or some other bound?
- How do the new oracles impact multi-target attacks? E.g. Banegas and Bernstein [BB17].

Thank you.


Martin R Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner.
Ciphers for MPC and FHE.
In *EUROCRYPT 2015*. Springer, 2015.


Martin R Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner.
Ciphers for MPC and FHE.
Cryptology ePrint Archive, Report 2016/687, 2016.


Gustavo Banegas and Daniel J Bernstein.
Low-communication parallel quantum multi-target preimage search.
In *SAC 2017*. Springer, 2017.


Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp.
Tight bounds on quantum searching.
Fortschritte der Physik, 46(4-5):493–505, 1998.


Joan Boyar and Rene Peralta.
A depth-16 circuit for the AES s-box.
Cryptology ePrint Archive, Report 2011/332, 2011.
`http://eprint.iacr.org/2011/332`.


Itai Dinur, Daniel Kales, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger.
Linear equivalence of block ciphers with partial non-linear layers: Application to lowmc.

In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 343–372, Cham, 2019. Springer International Publishing.

Joan Daemen and Vincent Rijmen.
Specification for the advanced encryption standard (aes).
Federal Information Processing Standards Publication, 197, 2001.

Craig Gidney.
Halving the cost of quantum addition.
Quantum, 2:74, June 2018.

David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo.
An algorithm for the t-count.
Quantum Information & Computation, 14(15-16):1261–1276, 2014.

Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt.
Applying grover's algorithm to AES: quantum resource estimates.
In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2016.

Lov K. Grover.
A fast quantum mechanical algorithm for database search.
In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

Panjin Kim, Daewan Han, and Kyung Chul Jeong.

Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to aes and sha-2.
Quantum Information Processing, 17(12):339, Oct 2018.

Brandon Langenberg, Hai Pham, and Rainer Steinwandt.
Reducing the cost of implementing aes as a quantum circuit.
Cryptology ePrint Archive, Report 2019/854, 2019.
https://eprint.iacr.org/2019/854.

Alexander Maximov.
Aes mixcolumn with 92 xor gates.
Cryptology ePrint Archive, Report 2019/833, 2019.
https://eprint.iacr.org/2019/833.

Giulia Meuli, Mathias Soeken, Earl Campbell, Martin Roetteler, and Giovanni De Micheli.
The role of multiplicative complexity in compiling low t-count oracle circuits.
CoRR, abs/1908.01609, 2019.

Giulia Meuli, Mathias Soeken, Martin Roetteler, Nikolaj Bjørner, and Giovanni De Micheli.
Reversible pebbling game for quantum memory management.
In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*, pages 288–291, 2019.

National Institute of Standards and Technology.

Submission requirements and evaluation criteria for the Post-Quantum Cryptography standardization process.
http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf, December 2016.


Peter Selinger.
Quantum circuits of *t*-depth one.
Phys. Rev. A, 87:042302, Apr 2013.


L.N. Trefethen and D. Bau.
Numerical Linear Algebra.
Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1997.


Akihiro Yamamura and Hirokazu Ishizuka.
Quantum cryptanalysis of block ciphers (algebraic systems, formal languages and computations).
2000.


Christof Zalka.
Grover's quantum searching algorithm is optimal.
Phys. Rev. A, 60, 10 1999.


Fang Zhang and Jianxin Chen.
Optimizing t gates in clifford+t circuit as $\pi/4$ rotations around paulis.
arXiv preprint arXiv:1903.12456, 2019.

Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig.
Picnic.
Technical report, NIST, 2017.