Using Quantum Programming Languages for Cryptanalysis

Fernando Virdia

Information Security Group,
Royal Holloway, University of London,
United Kingdom

April 15, 2021

Overview

In this talk I'll showcase 'non-asymptotic' cryptanalysis. I will cover:

- The crypto setting,
- Applications,
- An example work on AES using Q#,
- What features were the most useful so far, and what new useful features could be.

Overview

In this talk I'll showcase 'non-asymptotic' cryptanalysis. I will cover:

- The crypto setting,
- Applications,
- An example work on AES using Q#,
- What features were the most useful so far, and what new useful features could be.

Disclaimer

My personal experience covers mostly Q#, with very little use of Qiskit and ProjectQ.

Cryptography

The study of techniques to provide confidentiality, integrity and authenticity to communications.

Cryptanalysis

The study of ways to defeat cryptographic security measures.

Cryptography

The study of techniques to provide confidentiality, integrity and authenticity to communications.

Cryptanalysis

The study of ways to defeat cryptographic security measures.

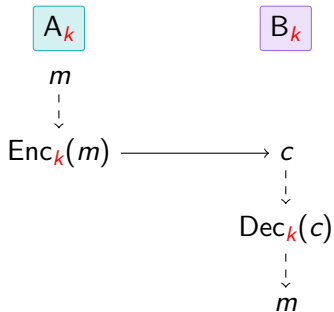
- Cryptographic functionality gets its security from computationally "hard" problems.
- Cryptanalysis allows us to determine how small we can choose problem instances, while achieving a target security level.

We will talk about two kinds of encryption primitives.

We will talk about two kinds of encryption primitives.

Block ciphers (eg. AES)

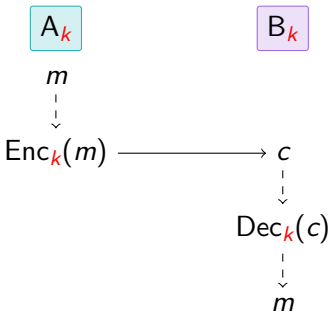
Two parties share an identical *key* k.



We will talk about two kinds of encryption primitives.

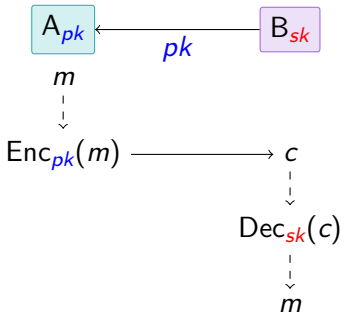
Block ciphers (eg. AES)

Two parties share an identical *key k*.



Public-key encryption (eg. RSA)

Different keys for sender and receiver.



Quantum cryptanalysis

We said security depends on computational hardness. Quantum computing will have an effect on this picture.

Quantum cryptanalysis

We said security depends on computational hardness. Quantum computing will have an effect on this picture.

Shor's algorithm

Polynomial time algorithm for solving RSA.

$\Longrightarrow$ Need for "post-quantum" cryptography, that resists quantum advesaries.

Quantum cryptanalysis

We said security depends on computational hardness. Quantum computing will have an effect on this picture.

Shor's algorithm

Polynomial time algorithm for solving RSA.

$\Longrightarrow$ Need for "post-quantum" cryptography, that resists quantum advesaries.

Grover's algorithm

Quadratic speed up to generic exhaustive key-search attack.

$\Longrightarrow$ Asymptotic solution: double key lengths, but we can do better.

Quantum cryptanalysis

We said security depends on computational hardness. Quantum computing will have an effect on this picture.

Shor's algorithm

Polynomial time algorithm for solving RSA.

$\Longrightarrow$ Need for "post-quantum" cryptography, that resists quantum advesaries.

Grover's algorithm

Quadratic speed up to generic exhaustive key-search attack.

$\Longrightarrow$ Asymptotic solution: double key lengths, but we can do better.

We will now look at a non-asymptotic Grover scenario, using AES as an example.

(N, M)-unstructured search problem

Given a randomly sorted list L of size N and a property P such that exactly M elements of L satisfy P, find one such element.

(N, M)-unstructured search problem

Given a randomly sorted list L of size N and a property P such that exactly M elements of L satisfy P, find one such element.

$\Longrightarrow$ Classically this requires $O(N)$ steps, Grover's solves it in $O(\sqrt{N})$ steps.

(N, M)-unstructured search problem

Given a randomly sorted list L of size N and a property P such that exactly M elements of L satisfy P, find one such element.

$\implies$ Classically this requires $O(N)$ steps, Grover's solves it in $O(\sqrt{N})$ steps.

AES block cipher

Block cipher with encryption function E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.
$E(\cdot, m)$ considered indistinguishable from a random function over $\{0,1\}^k \mapsto \{0,1\}^n$.

Attacking AES: given (m, c), find k such that $c \leftarrow E(k, m)$.

(N, M)-unstructured search problem

Given a randomly sorted list L of size N and a property P such that exactly M elements of L satisfy P, find one such element.

$\implies$ Classically this requires $O(N)$ steps, Grover's solves it in $O(\sqrt{N})$ steps.

AES block cipher

Block cipher with encryption function E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.
$E(\cdot, m)$ considered indistinguishable from a random function over $\{0,1\}^k \mapsto \{0,1\}^n$.

Attacking AES: given (m, c), find k such that $c \leftarrow E(k, m)$.

Since $E(\cdot, m) \sim \$$, this is an unstructured search in $\{0,1\}^k$.

$\implies$ Classical runtime $\approx 2^k$ encryptions, one per key (trivially parallelisable).

$\implies$ Quantum runtime $\approx 2^{k/2}$ Grover steps (badly parallelisable).

Asymptotically, we are "done" with cryptanalysis: 2^k vs $2^{k/2}$ means doubling the key length k is enough.

Why attempt a non-asymptotic cryptanalysis?

Asymptotically, we are "done" with cryptanalysis: 2^k vs $2^{k/2}$ means doubling the key length k is enough.

Why attempt a non-asymptotic cryptanalysis?

- General reason: doubling keys may be practically inconvenient (and overkill).
- Particular reason: the hardness of AES is being used as a definition of security.

Asymptotically, we are "done" with cryptanalysis: 2^k vs $2^{k/2}$ means doubling the key length k is enough.

Why attempt a non-asymptotic cryptanalysis?

- General reason: doubling keys may be practically inconvenient (and overkill).
- Particular reason: the hardness of AES is being used as a definition of security.

NIST Post-Quantum Cryptography standardisation

- Since 2017, the US NIST has been running a process to standardise post-quantum public-key cryptographic schemes.
- To qualify for "category 5" security, a scheme should be as secure as AES-256.
- Extra constraint: upper bound `MAXDEPTH` on the depth of quantum computation.

Post-quantum cryptography

NIST wants to standardise new *public-key encryption* and *signature schemes* with:

- Well-understood hardness against quantum adversaries,
- Smallest possible size.

Post-quantum cryptography

NIST wants to standardise new *public-key encryption* and *signature schemes* with:

- Well-understood hardness against quantum adversaries,
- Smallest possible size.

A practical analysis of Grover's algorithm is also useful here.

- Most quantum attacks on the proposals are classical attacks using Grover speed-ups for searching steps.

NIST concerns attacking AES-256

Parallelisation

NIST considers a hard limit $\texttt{MAXDEPTH} \in \{2^{40}, 2^{64}, 2^{96}\}$.
$\texttt{MAXDEPTH} < 2^{k/2} = 2^{128}$, what is naively required by Grover's
$\implies$ We need to account for Grover's parallelisation.

NIST concerns attacking AES-256

Parallelisation

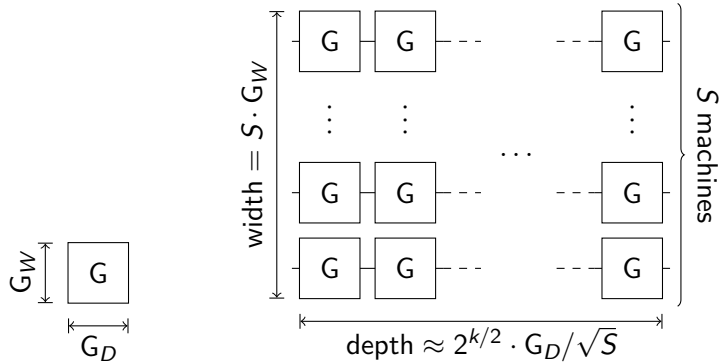
NIST considers a hard limit $\texttt{MAXDEPTH} \in \{2^{40}, 2^{64}, 2^{96}\}$.
$\texttt{MAXDEPTH} < 2^{k/2} = 2^{128}$, what is naively required by Grover's
$\implies$ We need to account for Grover's parallelisation.

Resource estimation

NIST further proposes the measure of security being the number of quantum gates required for the attack.
$\implies$ How many gates is $O(2^{k/2})$ Grover steps?

Parallel Grover

Grover parallelises badly [Zal99]. Rule of thumb, using S machines gives a $\sqrt{S}$ speed-up. For a fully analysis in our setting, see [JNRV20].



(a) Grover step (b) Parallel Grover circuit

$\Longrightarrow$ To derive S we need to know the size of G.

Resource estimation

The main component of G is U_f: $|x\rangle\,|\varphi\rangle \mapsto (-1)^{f(x)}\,|x\rangle\,|\varphi\rangle$, where f marks solutions to the search problem.

In our case, $f(k; m, c) = [\![E_k(m) = c]\!] \implies U_f$ implements AES encryption.

Resource estimation

The main component of G is U_f: $|x\rangle |\varphi\rangle \mapsto (-1)^{f(x)} |x\rangle |\varphi\rangle$, where f marks solutions to the search problem.

In our case, $f(k; m, c) = [\![E_k(m) = c]\!] \implies U_f$ implements AES encryption.

We are interested in finding

- G's width: the number of qubits used,
- G's depth: the depth of the circuit in terms of gates,
- G's gates: the total number of gates used in G.

Resource estimation

The main component of G is U_f: $|x\rangle\,|\varphi\rangle \mapsto (-1)^{f(x)}\,|x\rangle\,|\varphi\rangle$, where f marks solutions to the search problem.

In our case, $f(k; m, c) = [\![E_k(m) = c]\!] \implies U_f$ implements AES encryption.

We are interested in finding

- G's width: the number of qubits used,
- G's depth: the depth of the circuit in terms of gates,
- G's gates: the total number of gates used in G.

Strategy

Implement G (hence AES) in Q#, use the resource estimator to find circuit size.

Advantages vs manual circuit design

- partially unit-testable,
- friendly to read/modify,
- automated circuit size estimates,
- easy to port existing linear programs/verilog code.

Advantages vs manual circuit design

- partially unit-testable,
- friendly to read/modify,
- automated circuit size estimates,
- easy to port existing linear programs/verilog code.

Requires a computational model

- We only work with logical qubits,
- No costs for idle qubits,
- No need for gates to operate only locally,
- Swapping qubits is free, by rewiring.

Advantages vs manual circuit design

- partially unit-testable,
- friendly to read/modify,
- automated circuit size estimates,
- easy to port existing linear programs/verilog code.

Requires a computational model

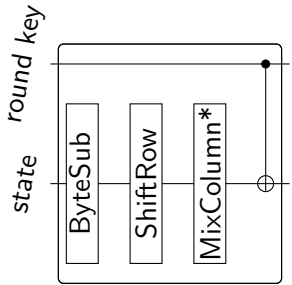
- We only work with logical qubits,
- No costs for idle qubits,
- No need for gates to operate only locally,
- Swapping qubits is free, by rewiring.

Let's now look at the components of AES.

- AES is a substitution-permutation network (SPN).
- Two parallel processes, one updates a round key, the other operates on the message block. Both operate in repeated *rounds*.

Credit for key expansion: Jérémy Jean.

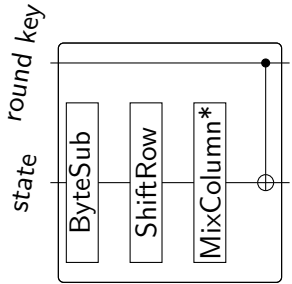
- AES is a substitution-permutation network (SPN).
- Two parallel processes, one updates a round key, the other operates on the message block. Both operate in repeated *rounds*.



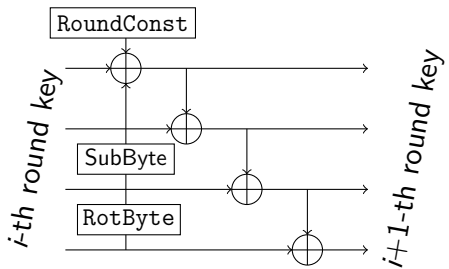
(e) AES encryption round

Credit for key expansion: Jérémy Jean.

- AES is a substitution-permutation network (SPN).
- Two parallel processes, one updates a round key, the other operates on the message block. Both operate in repeated *rounds*.



(g) AES encryption round



(h) AES (-128) key expansion round

Credit for key expansion: Jérémy Jean.

Component	Mathematical operation	Quantum gates
ShiftRow	Permutation	Rewiring
RotByte	Permutation	Rewiring
RoundConst	Fixed bit-flip	X
MixColumn	Invertible affine map $\mathbb{F}_2^{32}$	CNOT
SubByte, ByteSub	Inversion + affine maps over $\mathbb{F}_{2^8}$	NOT, XOR, AND/CCNOT

Component	Mathematical operation	Quantum gates
ShiftRow	Permutation	Rewiring
RotByte	Permutation	Rewiring
RoundConst	Fixed bit-flip	X
MixColumn	Invertible affine map $\mathbb{F}_2^{32}$	CNOT
SubByte, ByteSub	Inversion + affine maps over $\mathbb{F}_{2^8}$	NOT, XOR, AND/CCNOT

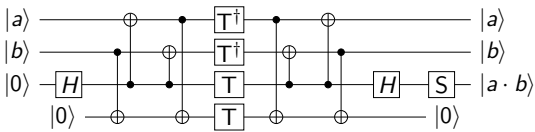
There are a few options for generating linear and non-linear maps:

- LU matrix decomposition allows in-place invertible maps, no need for ancillas,
- Porting linear programs from the classical hardware design by implementing $(x, y, z) \mapsto (x, y, z \oplus (x \star y))$ for any binary operation $\star$,
- Using SAT-based synthetisers (eg. [FSK10, MSR+19, MSC+19]).

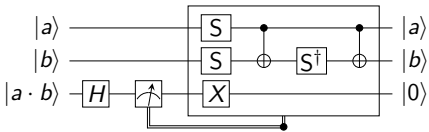
There is also room for choosing different binary operation circuits. For example, we can use AND instead of CCNOT.

We thank Mathias Soeken for providing the implementation of the AND gate circuit.

There is also room for choosing different binary operation circuits. For example, we can use AND instead of CCNOT.



(k) AND gate.



(l) AND† gate.

We use a combination of Selinger's [Sel13] and Jones' [Jon13] circuits to express the AND gate in terms of Clifford and T gates.

$\Longrightarrow$ It requires expressing conditional branching on measurement outputs.

We thank Mathias Soeken for providing the implementation of the AND gate circuit.

What are the results?

Slightly smaller numbers have since been obtained in the same computational model.

What are the results?

We said asymptotically the quantum cost is $\approx 2^{128}$, the classical cost $\approx 2^{256}$.

MAXDEPTH	# of parallel machines	# of gates
2^{40}	$2^{197.15}$	$2^{245.46}$
2^{64}	$2^{149.15}$	$2^{221.46}$
2^{96}	$2^{85.17}$	$2^{190.47}$

Table: Derived gate costs for attacking AES-256 using Grover's algorithm.

Slightly smaller numbers have since been obtained in the same computational model.

What are the results?

We said asymptotically the quantum cost is $\approx 2^{128}$, the classical cost $\approx 2^{256}$.

MAXDEPTH	# of parallel machines	# of gates
2^{40}	$2^{197.15}$	$2^{245.46}$
2^{64}	$2^{149.15}$	$2^{221.46}$
2^{96}	$2^{85.17}$	$2^{190.47}$

Table: Derived gate costs for attacking AES-256 using Grover's algorithm.

$\Longrightarrow$ Doubling keys may not be necessary.
$\Longrightarrow$ Quantum speed-ups with depth limit not as dramatic for symmetric crypto.

Slightly smaller numbers have since been obtained in the same computational model.

We've seen one approach to non-asymptotic quantum cryptanalysis.

- How popular is it? What languages are commonly used?
- What language features are or would be useful?

Q#	[JNRV20] AES
	[HJN+20] Shor's on elliptic curves
	[BJ20] Simon's against various block ciphers
	[DP21] Search With Two Oracles technique for AES
Qiskit	[AMM20a] SIMON
	[CS20] ARIA
	[Sch20] Gimli
	[AMM+20b] FSR-based constructions
ProjectQ	[LPS20] AES S-box
	[JCKS20] SPECK
	[JCK+20] Korean block ciphers
IBMQ	[MSS21] Simon's algorithm (on `ibmq_16_melbourne`)
Other	[GLRS16] AES (in C)
	[RNSL17] Shor's on elliptic curves (in LIQUi|⟩)
	[AGPS20] Nearest-Neighbour Search for lattice sieving (in Python)

Table: Selection of works implementing cryptanalytic quantum algorithms.

Quantum Period Finding on IBM-Q

May *et al.* run period finding on $\mathbb{F}_2^7 \mapsto \mathbb{F}_2^7$ using `ibmq_16_melbourne`.

Approach

- Manually optimise circuit compatible with computer's topology.
- Apply various noise-smoothing techniques to address qubit quality and "bias towards 0".
- Recovering the period on NISQ is reduced to Learning Parity with Noise (LPN).

With constant error-rate τ, quantum runtime $2^{O(\frac{n}{\log(n/\tau)})}$ vs $2^{n/2}$ classical runtime.

Useful and desired features

Maybe the most useful feature while implementing:

Full cheap simulation of classical sub-circuits

- Q# exposes this via its "ToffoliSimulator", and allowed us to unit-test our circuits.
- I wrote a naive implementation for Qiskit for testing the AES S-box, at `https://github.com/fvirdia/Qiskit-DIY-Toffoli-simulator`.

Interesting functionality would be having a noise and error correction simulators.

Second most useful feature:

Language expressiveness

- Classical looping and branching notation, even conditioned on measurement output.
- Flexible qubit allocation (automatic and manual).
- Utilities such as “auto adjoint” computation.
- Both circuit-level and time-evolution descriptions.

Being able to compile to circuits is great.

Ok, I lied, this is the most useful feature:

Automatic resource estimation

- Including branches and measurements-conditioned components.
- Ideally with a mode for including error correction cost and imposing related constraints (like non-local qubit operations).

Something we definitely missed:

(Some) automatic circuit optimization

- Optimising qubit allocation (with min-depth vs min-width options).
- Automatic linear map circuits (with min-depth vs min-width vs min-gates options).
- "Quantum-exclusive" optimisations would be nice [GKMR14, ZC19].
- If an optimisation across scopes/function boundaries is found, report it clearly.

Something else we missed.

Documentation with examples

- We found lots of documentation consisted of function references without examples.
- I love the Qiskit textbook.

Ideally, this pairs with openly accessible bug tracking!

In Conclusion

- Cryptographers are adopting quantum programming languages.
- The use case presents different constraints.
- Some work also in the NISQ setting.

Thank you


Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck.
Estimating quantum speedups for lattice sieves.
In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 583–613, Cham, 2020. Springer International Publishing.


Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay.
Grover on SIMON.
Quantum Information Processing, 19(9):340, Sep 2020.


Ravi Anand, Subhamoy Maitra, Arpita Maitra, Chandra Sekhar Mukherjee, and Sourav Mukhopadhyay.
Resource estimation of grovers-kind quantum cryptanalysis against FSR based symmetric ciphers.
Cryptology ePrint Archive, Report 2020/1438, 2020.
https://eprint.iacr.org/2020/1438.


Xavier Bonnetain and Samuel Jaques.
Quantum period finding against symmetric primitives in practice.
Cryptology ePrint Archive, Report 2020/1418, 2020.
https://eprint.iacr.org/2020/1418.


Amit Kumar Chauhan and Somitra Kumar Sanadhya.
Quantum resource estimates of grover's key search on aria.
In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 238–258, Cham, 2020. Springer International Publishing.


James H Davenport and Benjamin Pring.
Improvements to quantum search techniques for block-ciphers, with applications to AES.

In *Selected Areas in Cryptography – SAC 2020*, 2021.


Carsten Fuhs and Peter Schneider-Kamp.
Synthesizing shortest linear straight-line programs over gf (2) using sat.
In *International Conference on Theory and Applications of Satisfiability Testing*, pages 71–84. Springer, 2010.


David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo.
An algorithm for the T-Count.
Quantum Info. Comput., 14(15–16):1261–1276, November 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*, pages 29–43. Springer, Heidelberg, 2016.


Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken.
Improved quantum circuits for elliptic curve discrete logarithms.
In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 425–444. Springer, Heidelberg, 2020.


Kyoungbae Jang, Seungju Choi, Hyeokdong Kwon, Hyunji Kim, Jaehoon Park, and Hwajeong Seo.
Grover on Korean block ciphers.
Applied Sciences, 10(18), 2020.


Kyungbae Jang, Seungjoo Choi, Hyeokdong Kwon, and Hwajeong Seo.
Grover on SPECK: Quantum resource estimates.

Cryptology ePrint Archive, Report 2020/640, 2020.
https://eprint.iacr.org/2020/640.


Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia.
Implementing grover oracles for quantum key search on AES and LowMC.
In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 280–310. Springer, Heidelberg, May 2020.


Cody Jones.
Low-overhead constructions for the fault-tolerant Toffoli gate.
Phys. Rev. A, 87:022328, Feb 2013.


Brandon Langenberg, Hai Pham, and Rainer Steinwandt.
Reducing the cost of implementing the Advanced Encryption Standard as a quantum circuit.
IEEE Transactions on Quantum Engineering, 1:1–12, 2020.


Giulia Meuli, Mathias Soeken, Earl Campbell, Martin Roetteler, and Giovanni De Micheli.
The role of multiplicative complexity in compiling low *t*-count oracle circuits.
In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.


Giulia Meuli, Mathias Soeken, Martin Roetteler, Nikolaj Bjorner, and Giovanni De Micheli.
Reversible pebbling game for quantum memory management.
In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 288–291. IEEE, 2019.


Alexander May, Lars Schlieper, and Jonathan Schwinger.
Noisy Simon period finding.

In *Cryptographers' Track at the RSA Conference*. Springer, 2021.


Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin E. Lauter.
Quantum resource estimates for computing elliptic curve discrete logarithms.
In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 241–270. Springer, Heidelberg, December 2017.


Lars Schlieper.
In-place implementation of quantum-Gimli.
arXiv preprint arXiv:2007.06319, 2020.


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


Christof Zalka.
Grover's quantum searching algorithm is optimal.
Phys. Rev. A, 60:2746–2751, Oct 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.