SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
Special Focus on Quantum Information

October 2022, Vol. 65 200503:1–200503:12
https://doi.org/10.1007/s11432-022-3511-5

Variational quantum attacks threaten advanced encryption standard based symmetric cryptography

Zeguo WANG[1,2], Shijie WEI[2,1*], Gui-Lu LONG[1,2,3,4*] & Lajos HANZO[5]

[1]*State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing* 100084*, China;*
[2]*Beijing Academy of Quantum Information Sciences, Beijing* 100193*, China;*
[3]*Beijing National Research Center for Information Science and Technology, School of Information, Tsinghua University, Beijing* 100084*, China;*
[4]*Frontier Science Center for Quantum Information, Beijing* 100193*, China;*
[5]*Department of Electronics and Computer Science (ECS), University of Southampton, Southampton* SO17 1BJ*, UK*

Received 11 April 2022/Revised 1 June 2022/Accepted 14 June 2022/Published online 18 July 2022

Abstract We propose a variational quantum attack algorithm (VQAA) for classical advanced encryption standard (AES)-like symmetric cryptography, as exemplified by the simplified-data encryption standard (S-DES). In the VQAA, the known ciphertext is encoded as the ground state of a Hamiltonian that is constructed through a regular graph, and the ground state can be found using a variational approach. We designed the ansatz and cost function for the S-DES's variational quantum attack. It is surprising that sometimes the VQAA is even faster than Grover's algorithm as demonstrated by our simulation results. The relationships of the entanglement entropy, concurrence, and the cost function are investigated, which indicate that entanglement plays a crucial role in the speedup.

Keywords S-DES, VQA, ansatz, cost function, optimization

Citation Wang Z G, Wei S J, Long G-L, et al. Variational quantum attacks threaten advanced encryption standard based symmetric cryptography. Sci China Inf Sci, 2022, 65(10): 200503, https://doi.org/10.1007/s11432-022-3511-5

1 Introduction

The security of information plays an important role in defense, in the economy and in people's livelihood [1, 2]. At the time of writing typically asymmetric cryptography, such as RSA that is proposed by Rivest, Shamir, and Adleman in 1977 [3], is used for transmitting the secret key, and symmetric cryptography, such as the advanced encryption standard (AES) [4], is employed for encrypting data. With the development of quantum computers [5–8], more and more attention is paid to the security analysis of classical cryptography under quantum attacks.

Shor's algorithm [9] is capable of decrypting RSA cryptography in polynomial time [10], which seriously threatens the security of asymmetric cryptography. For symmetric cryptography, Grover's algorithm [11–13] can find the key in a set having N entries by only evaluating on the order of $\sqrt{N}$ entries. In [14–17], the efficient quantum implementations of both the AES and of the data encryption standard (DES) are proposed by relying on less quantum resources, such as qubits, quantum gates, and circuit depths.

At the time of writing, we are in the noisy intermediate-scale quantum (NISQ) era [18] when quantum computing systems are characterized by the low number of qubits, low fidelity, and shallow quantum circuits. Under these restrictions, various classical-quantum hybrid algorithms, including the variational quantum algorithm (VQA) [19–22], and the quantum approximate optimization algorithm (QAOA) [23] have been proposed. These hybrid algorithms have significant advantages in solving combinatorial optimization [24] and Hamiltonian ground state problems [25]. Briefly, the VQA has found applications both in quantum chemistry [26, 27], as well as in quantum machine learning [28–30], and in quantum

* Corresponding author (email: weisj@baqis.ac.cn, gllong@tsinghua.edu.cn)

finance [31–33]. However, there is a paucity of research on the employment of VQA in classical symmetric cryptography attacks. We fill this knowledge-gap by conceiving a quantum attack scheme based on the VQA for AES-like symmetric cryptography. In our design, the parameterized quantum circuit (PQC) operates on the key space, and the cost function is designed according to the known ciphertext. We will show by our simulations that the variational quantum attack algorithm (VQAA) on average uses the same order of search-space queries as Grover's algorithm. However, in some cases, it is even faster than Grover's algorithm, which is really surprising. We also investigated the relationship between the entanglement entropy, concurrence, and the cost function. It was found that the speedup attained is related to the entropy, which is not unexpected, because the entropy by definition represents the specific degree of surprise upon revealing a particular problem solution/outcome.

The paper is organized as follows. Section 2 briefly reviews the structure of symmetric cryptography and of the simplified-DES (S-DES) technique. Section 3 describes the VQAA process in our work. In this part, the cost function, the ansatzes, and the classical optimization algorithms are designed. Then, the optimization results and the entanglement entropy, as well as the associated concurrence, are presented and analyzed. Finally, a summary is provided.

2 Symmetric cryptography

The symmetric cryptography, including AES and DES, encrypts and decrypts the data using the same key. For AES [4], there are key four operations namely, AddRoundKey, SubBytes, ShiftRows, and MixColumns, to replace and substitute the data. For DES [34], the encryption mainly includes the IP operation, the f function, and the SWAP operation. We use the S-DES as an example for characterizing the VQAA for simplicity. The specific process of S-DES is as follows.

The input of S-DES is an 8-bit plaintext and a 10-bit key. The output is an 8-bit ciphertext. We can simply express the encryption process as a composite of the following functions:

$$\text{Ciphertext} = \text{IP}^{-1} \circ f_{K_2} \circ \text{SW} \circ f_{K_1} \circ \text{IP}[\text{Plaintext}], \tag{1}$$

where IP is the initial replacement, the function f_K includes the replacement and substitution operations, SW is a SWAP function, and K_1, K_2 are the sub-keys in the first and second round of encryption, respectively.

The decryption represents the inversion of encryption in the form of

$$\text{Plaintext} = \text{IP}^{-1} \circ f_{K_1} \circ \text{SW} \circ f_{K_2} \circ \text{IP}[\text{Ciphertext}]. \tag{2}$$

The encryption process is shown in Figure 1, and the associated details are given in Appendix A.

3 VQAA for symmetric cryptography

The main idea of our VQAA is shown in Figure 2. Based on a pair of known ciphertext and plaintext, the associated Hamiltonian is designed, whose ground state is the ciphertext. The parameterized quantum circuit gives a linear combination of all possible keys. After the symmetric cryptography operations, we have a linear combination of all the ciphertext corresponding to the known plaintext, associated with all possible keys. Then the variational process is started to find the Hamiltonian associated with the lowest energy, which contains the corresponding key.

We now demonstrate how to carry out a quantum attack on the S-DES using the VQAA as an example. The plaintext and ciphertext are represented by 8-bit quantum states. Firstly, we construct the Hamiltonian whose ground state corresponds to the ciphertext. The detailed construction of the Hamiltonian is assumed to be given. Secondly, the key space is encoded into an adjustable quantum state by a parameterized quantum circuit which is also known as ansatz. Next, the output of the parameterized quantum circuit is used as a key to encrypt the known plaintext based on the S-DES, and then the superposition of ciphertexts is obtained. Finally, we measure the superposition of ciphertexts and forward the result to the classical optimization algorithm. By using the optimization algorithm, we adjust the input parameters of the parameterized quantum circuit to arrange for the superposition ciphertext state to have a considerable overlap with the known ciphertext. When the result of measurement is

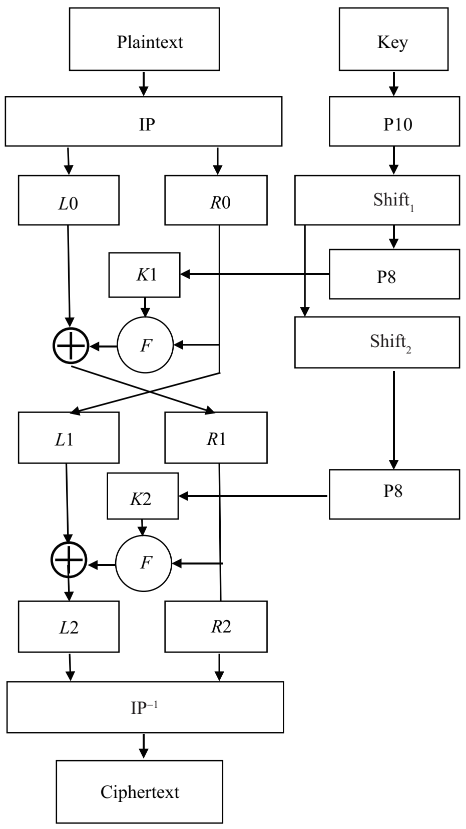


Figure 1 The encryption process of S-DES.

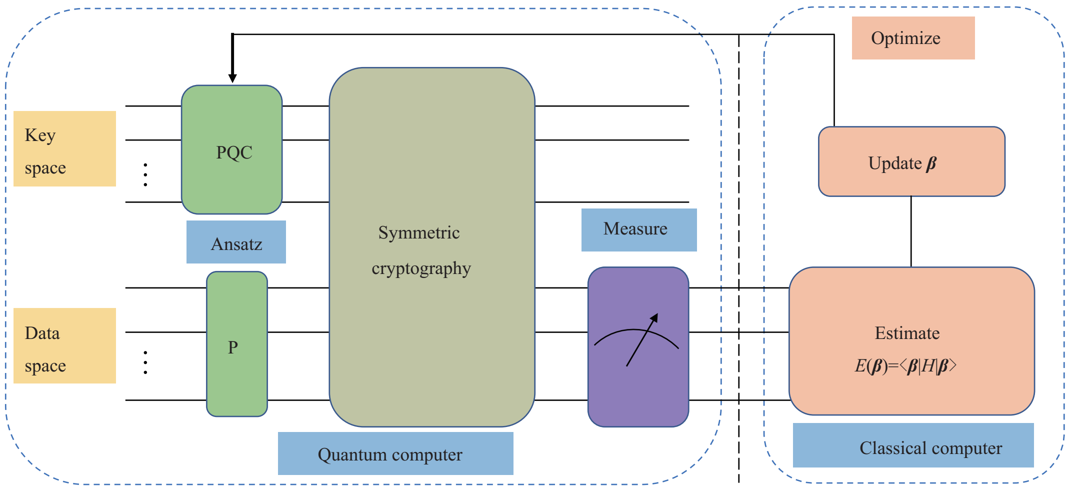


Figure 2 (Color online) Schematic diagram of the quantum attack on S-DES using VQAs, where $\boldsymbol{\beta}$ represents the parameters of PQC.

the known ciphertext, the key space also collapses to the required key state. In the VQAA of S-DES, the key space and the data space contain 10-qubit and 8-qubit strings, respectively, and the 'symmetric cryptography' block of Figure 2 is substituted by the 'S-DES' module, whose quantum implementation can be found in [17].

3.1 The construction of cost function

In order to encode the known ciphertext into a Hamiltonian ground state, we use each bit as a node to construct regular graphs. For an 8-node network, we can construct an n-regular $(n = 1, 2, \ldots, 7)$ graph. The value of the i-th node is denoted by $V(i)$, which is the value of the i-th bit. If there is a pair of

Table 1 The energy range of seven Hamiltonians, where 'reg' represents regular

	1-reg	2-reg	3-reg	4-reg	5-reg	6-reg	7-reg
Ground energy	-8	-12	-16	-20	-24	-28	-32
The highest energy	4	8	8	9	12	8	4
The first excited energy	-6	-7	-9	-12	-16	-20	-24
Ratio	0.1667	0.2500	0.2917	0.2759	0.2222	0.2222	0.2222

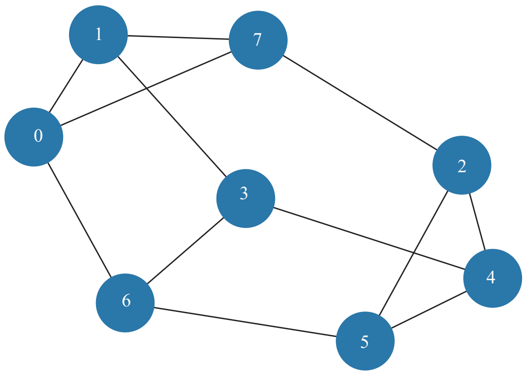


Figure 3 (Color online) The 3-regular graph used in Hamiltonian construction.

nodes i and j in the graph that are connected, we add the term $w_{ij}Z_iZ_j$ into the Hamiltonian, where Z is the Pauli-Z operator, $i, j \in \{0, 1, \ldots, 7\}$. The coefficient w_{ij} is determined by $V(i)$ and $V(j)$, which takes the form

$$w_{ij} = \begin{cases} 1, & \text{if } V(i) \neq V(j), \\ -1, & \text{if } V(i) = V(j). \end{cases} \tag{3}$$

Additionally, all of the single-qubit operators $\sum_{i=0}^{7} t_iZ_i$ are added into the Hamiltonian, where t_i is defined as

$$t_i = \begin{cases} 0.5, & \text{if } V(i) = 1, \\ -0.5, & \text{if } V(i) = 0. \end{cases} \tag{4}$$

Then the energy level of these seven Hamiltonians is analyzed. The results are shown in Table 1. The 'Ratio' represents the ratio of the energy level differences between the ground state and the first excited state to the total dynamic range of the energy levels.

Instinctively, the higher the ratio, the easier it is to distinguish the global minimum. Because a large ratio implies that the gradient changes rapidly in the vicinity of the minimum value, the optimization path tends to lean more toward the neighborhood of the minimum. The simulations in Appendix B prove our prediction. The optimization works best when $n = 3$. The 3-regular graph we use is shown in Figure 3. The number of items in the Hamiltonian is $3 \times 8/2 + 8 = 20$, which is a polynomial whose order is determined by the number of nodes.

The Hamiltonian defined in Figure 3 is

$$\begin{aligned} H = {} & w_{01}Z_0Z_1 + w_{06}Z_0Z_6 + w_{07}Z_0Z_7 + w_{13}Z_1Z_3 + w_{17}Z_1Z_7 \\ & + w_{24}Z_2Z_4 + w_{25}Z_2Z_5 + w_{27}Z_2Z_7 + w_{34}Z_3Z_4 + w_{36}Z_3Z_6 \\ & + w_{45}Z_4Z_5 + w_{56}Z_5Z_6 + \sum_{i=0}^{7} t_iZ_i. \end{aligned} \tag{5}$$

As described earlier, w_{ij} and t_i depend on the ciphertext. The cost function $E(\boldsymbol{\beta})$ is the expectation of the Hamiltonian,

$$E(\boldsymbol{\beta}) = \langle \boldsymbol{\beta} | H | \boldsymbol{\beta} \rangle, \tag{6}$$

where $|\boldsymbol{\beta}\rangle$ is the superposition of ciphertext state.

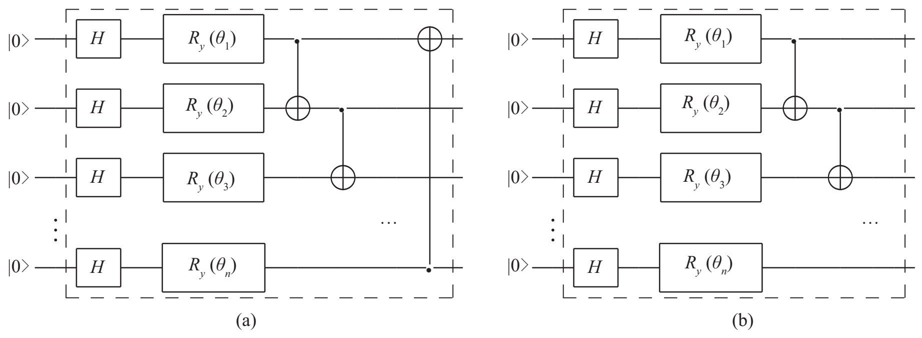


Figure 4 The dashed box indicates a single Y-Cx circuit layer that can be repeated. (a) and (b) represent two kinds of Y-Cx models. The only difference is that (a) contains a CNOT gate from the last qubit to the first qubit.

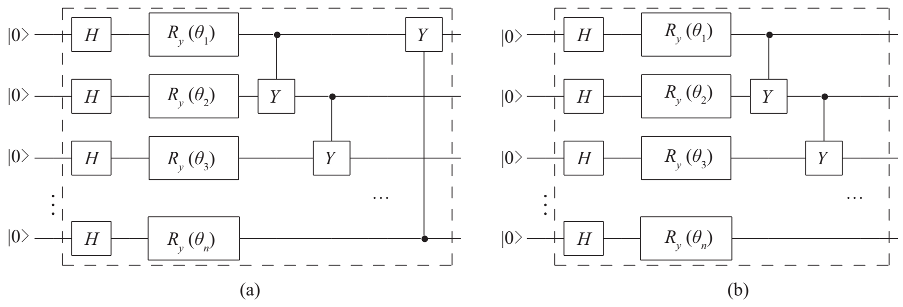


Figure 5 The dashed box indicates a single Y-Cy circuit layer that can be repeated. Gate Y represents a Pauli-Y gate: (a) and (b) represent two kinds of Y-Cy models. The only difference is that (a) contains a controlled Y gate from the last qubit to the first qubit.

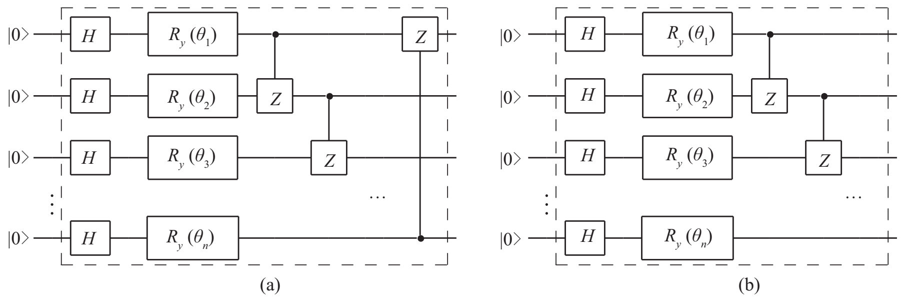
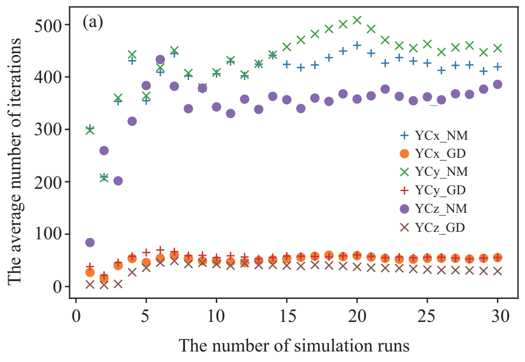


Figure 6 The dashed box indicates a single Y-Cz circuit layer that can be repeated. Gate Z represents a Pauli-Z gate: (a) and (b) represent two kinds of Y-Cy models. The only difference is that (a) contains a controlled Z gate from the last qubit to the first qubit.

3.2 Ansatz

We have chosen six ansatzes in this work. The first two are denoted as Y-Cx models, which are shown in Figure 4. The next two are denoted as Y-Cy models, which are shown in Figure 5. The last two are denoted as Y-Cz models, which are shown in Figure 6. The six ansatzes can be divided into two categories, as shown in Figures 4(a), 5(a), and 6(a), Figures 4(b), 5(b), and 6(b), respectively. They are denoted as A-ansatz and B-ansatz. Their differences are that there is a controlled X, Y, Z gate at the right-most edge of Figures 4(a), 5(a), and 6(a).

For an n-qubit system, a 1-layer ansatz requires n parameters, and their circuit depths are $n+2$ for an A-ansatz, and $n+1$ for a B-ansatz. The key space in S-DES contains 10 qubits, so it requires 10 parameters as its input for a 1-layer ansatz, and its circuit depth is 12 or 11 for A or B, respectively, which is far lower than the depth of S-DES's quantum implementation circuit. The initial state is prepared as the uniform superposition state.



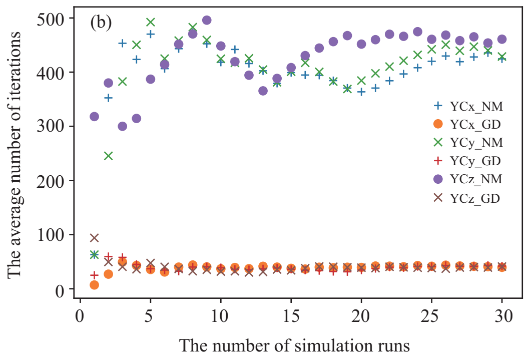


Figure 7 (Color online) The simulation results. (a) A-ansatz: the average number of iterations over the number of simulations; (b) B-ansatz: the average number of estimations over the number of simulations.

Table 2 The number of iterations in the A-ansatz (A) and B-ansatz (B), respectively

		N-M			GD		
		Maximum	Minimum	Average	Maximum	Minimum	Average
A	Y-Cx	694	28	419.40	94	2	55.27
	Y-Cy	691	28	454.93	94	3	55.67
	Y-Cz	692	21	385.83	94	2	29.50
B	Y-Cx	705	63	424.53	94	4	39.63
	Y-Cy	687	63	428.83	94	3	41.93
	Y-Cz	700	19	460.83	94	2	41.03

3.3 Classical optimization algorithms

We use two methods to optimize the parameters, namely the gradient descent method and the Nelder-Mead (N-M) method [35] whose pseudo-codes and hyper-parameters are given in Appendix C. The cut-off condition is set as -9, which is the first excited energy. When the expectation of Hamiltonian is less than -9, the superposition cipher state has a large overlap with the known ciphertext (the ground state). When the measurement result is the known ciphertext, the key space collapses to the desired key state. Additionally, we have to set the restart condition for both two optimization algorithms. Explicitly, the VQAA will be restarted when the norm of the gradient is lower than 0.8 in the gradient descent method and when we have $f(x_N) - f(x_0) < 0.15$ for the N-M method. Furthermore, $f(x_N)$ and $f(x_0)$ are the maximum and the minimum expectation of the Hamiltonian from the $N+1$ points, respectively.

4 The optimization results

We now characterize the performance of the VQAA for the different combinations of ansatzes and optimization algorithms using numerical simulations. The relationships between the cost function and the entanglement entropy, as well as the concurrence, are presented.

4.1 The number of iterations

The hyper-parameters of the classical optimization algorithms are adjusted to the optimal value for the different ansatzes. The initial input parameters are the same in each simulation, in which we use different classical optimization methods in order to search for the ground state. In each simulation, the key and plaintext are chosen randomly at the same time, and the ciphertext is determined. The range of the key is $[0, 2^{10}-1]$, and the range of both the plaintext and the ciphertext is $[0, 2^8-1]$. All these values have to be converted into binary strings and then prepared as quantum states. We performed thirty simulations, each with six experiments corresponding to a combination of three ansatzes and two classical optimization algorithms. We terminated the process if we failed to find the key after 2^{10} measurements. We portray the average number of iterations in Figures 7(a) and (b) for A-ansatz and B-ansatz, respectively. For the sake of reference, we gave all the number of iterations averaged over 1 to 30 simulations.

In Table 2, we have given the maximum, the minimum, and the average number of iterations required for the six-ansatz and two classical optimization algorithms. It is apparent that the results for those

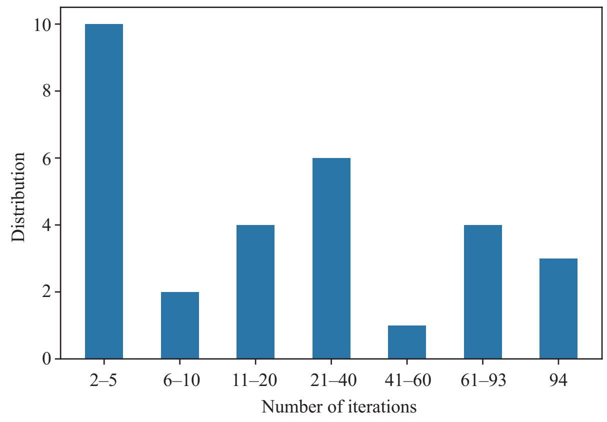


Figure 8 (Color online) The distribution of the number of iterations.

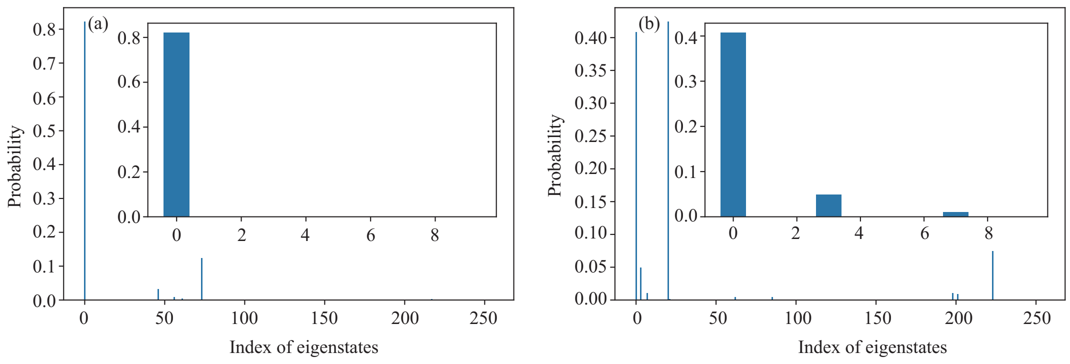


Figure 9 (Color online) The probability distribution of eigenstates for the Y-Cz(A) ansatz: (a) and (b) are two examples where the cost function value is lower than the threshold.

using the gradient descent method are much better than those using the N-M method. The gradient descent method usually takes 30–56 iterations to obtain the key, whereas the N-M method takes more than 400 iterations. For the six-ansatz scenario, the Y-Cz ansatz's results are better. Its average number of iterations is close to 32, which is the number of iterations of Grover's attack. At the same time, its minimum number of iterations is 2, which is better than Grover's algorithm. In Figure 8, we gave a distribution of iteration numbers for the A-ansatz with Y-Cz and gradient descent. It can be seen that in the range 2–5, there are ten times in total, accounting for one-third, which is a large proportion.

When the process is convergent, the occupation probability of the target state is the highest. Figure 9 presents the probability distribution of the eigenstates under the Y-Cz(A) ansatz, when the cost function value is lower than the threshold −9. The x-axis represents the eigenstates, which are ordered from the ground state to the highest eigenstate. The y-axis represents the corresponding probability. The probability of ground states in Figure 9 is 0.82 and 0.41, respectively.

4.2 Convergence vs. entanglement entropy/concurrence

Entanglement entropy and concurrence constitute a pair of popular entanglement metrics. In our work, the relationships between the cost function and the entanglement entropy as well as concurrence are investigated. For a pure state $\rho_{\mathrm{AB}} = |\Psi\rangle\langle\Psi|_{\mathrm{AB}}$, the entanglement entropy is defined as follows:

$$\mathcal{S}\left(\rho_{\mathrm{A}}\right) = -\operatorname{Tr}\left[\rho_{\mathrm{A}} \log \rho_{\mathrm{A}}\right] = -\operatorname{Tr}\left[\rho_{\mathrm{B}} \log \rho_{\mathrm{B}}\right] = \mathcal{S}\left(\rho_{\mathrm{B}}\right), \tag{7}$$

while the concurrence is defined as

$$\mathcal{C}_{\mathrm{A}}(\rho) = \sqrt{2\left(1 - \operatorname{Tr} \rho_{\mathrm{A}}^{2}\right)}, \tag{8}$$

where $\rho_{\mathrm{A}} = \operatorname{Tr}_{\mathrm{B}}\left(\rho_{\mathrm{AB}}\right)$ and $\rho_{\mathrm{B}} = \operatorname{Tr}_{\mathrm{A}}\left(\rho_{\mathrm{AB}}\right)$ represent the reduced density matrices for each partition.

We opted for the equal partition, where partition A represents the first 5 qubits and partition B the next 5 qubits, when calculating the entropy/concurrence.

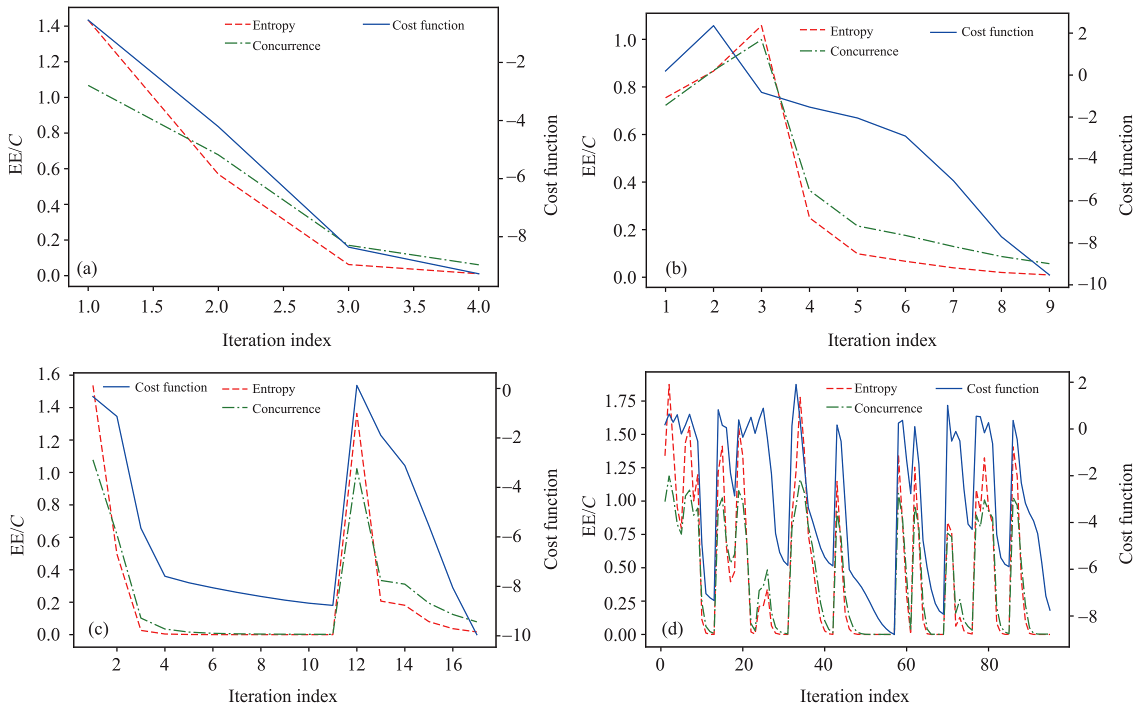


Figure 10 (Color online) The relationships between the entanglement entropy, concurrence, and the cost function, where the y-label EE and C represent the entanglement entropy and concurrence, respectively: (a)–(d) represent four different scenarios.

The results in Figure 10 are generated from the Y-Cz (A-ansatz) and the gradient descent method. There are four scenarios in Figure 10, and the y-label EE and C represent the entanglement entropy and concurrence, respectively.

(a) The process converges monotonically.

(b) Both EE & C first increase, and then gradually converge towards 0. This case represents a process where the search path is initially close to the target, but there is some initial divergence before convergence.

(c) The sudden spike represents a restart. As we recall, we have set up a limit for the gradient to be below, at which the iterations eventually are curtailed and then restarted again with a new set of parameters. Nevertheless, the process of searching for the target converges, and becomes successful.

(d) The process does not converge at all even after several restarts.

As we see from Figure 10, the entanglement entropy and concurrence behave similarly to the cost function. It indicates that entanglement is crucial for the success of VQAA.

5 Summary

In this work, we proposed a VQAA for orchestrating an attack in symmetric cryptography. We first constructed a cost function whose minimum corresponds to the Hamiltonian's ground state, which is the known ciphertext. The ground state is found by a VQA. Our simulations show that the gradient descent method is much faster than the N-M method. The result of gradient descent method is comparable to that of Grover's algorithm, and sometimes it is even faster than Grover's algorithm. It will be interesting to investigate further if this trend is also valid when the number of qubits becomes high. If this trend persists when the number of qubits is high, then it is a serious threat to the symmetric cryptography. There are still a lot of open issues, such as the employment of a better classical optimization algorithm, a better ansatz, and better initial parameters. It is worth noting that the security of the post-quantum algorithm under VQAA is also challenged. Because variational quantum algorithm does not have a fixed complexity, the VQAA proposed here may also applies to other computational complexity-based cryptographic algorithm. Further studies will continue in the future.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 12005015, 11974205, 11774197), National Key Research and Development Program of China (Grant No. 2017YFA0303700), Key Research and Development Program of Guangdong Province (Grant No. 2018B030325002), and Beijing Advanced Innovation Center for Future Chip (ICFC).

References

1 Feng D G, Lian Y F. Challenges to cyberspace security and countermeasures (in Chinese). Bull Chin Acad Sci, 2021, 36: 1239–1245
2 You X H, Wang C X, Huang J, et al. Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts. Sci China Inf Sci, 2021, 64: 110301
3 Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM, 1978, 21: 120–126
4 Joan D, Vincent R. The Design of Rijndael: AES-the Advanced Encryption Standard. Berlin: Springer, 2002
5 Arute F, Arya K, Babbush R, et al. Quantum supremacy using a programmable superconducting processor. Nature, 2019, 574: 505–510
6 Zhu Q L, Cao S R, Chen F S, et al. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. Sci Bull, 2022, 67: 240–245
7 Chang C R, Lin Y C, Chiu K L, et al. The second quantum revolution with quantum computers. AAPPS Bull, 2020, 30: 9–22
8 Kwek L C, Cao L, Luo W, et al. Chip-based quantum key distribution. AAPPS Bull, 2021, 31: 15
9 Shor P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev, 1999, 41: 303–332
10 Gidney C, EkeråM. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum, 2021, 5: 433
11 Grover L K. A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, 1996. 212–219
12 Long G L. Grover algorithm with zero theoretical failure rate. Phys Rev A, 2001, 64: 022307
13 Zhu Y Y, Wang Z G, Yan B, et al. Robust quantum search with uncertain number of target states. Entropy, 2021, 23: 1649
14 Grassl M, Langenberg B, Roetteler M, et al. Applying Grover's algorithm to AES: quantum resource estimates. In: Post-Quantum Cryptography. Berlin: Springer, 2016. 29–43
15 Zou J, Wei Z, Sun S, et al. Quantum circuit implementations of AES with fewer qubits. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, 2020. 697–726
16 Wang Z G, Wei S J, Long G L. A quantum circuit design of AES requiring fewer quantum qubits and gate operations. Front Phys, 2022, 17: 41501
17 Denisenko D V, Nikitenkova M V. Application of Grover's quantum algorithm for SDES key searching. J Exp Theor Phys, 2019, 128: 25–44
18 Preskill J. Quantum computing in the NISQ era and beyond. Quantum, 2018, 2: 79
19 Peruzzo A, McClean J, Shadbolt P, et al. A variational eigenvalue solver on a photonic quantum processor. Nat Commun, 2014, 5: 4213
20 Yung M H, Casanova J, Mezzacapo A, et al. From transistor to trapped-ion computers for quantum chemistry. Sci Rep, 2015, 4: 3589
21 Sung K J, Yao J, Harrigan M P, et al. Using models to improve optimizers for variational quantum algorithms. Quantum Sci Technol, 2020, 5: 044008
22 Cerezo M, Arrasmith A, Babbush R, et al. Variational quantum algorithms. Nat Rev Phys, 2021, 3: 625–644
23 Farhi E, Goldstone J, Gutmann S. A quantum approximate optimization algorithm. 2014. ArXiv:1411.4028
24 Harrigan M P, Sung K J, Neeley M, et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. Nat Phys, 2021, 17: 332–336
25 Cervera-Lierta A, Kottmann J S, Aspuru-Guzik A. Meta-variational quantum eigensolver: learning energy profiles of parameterized hamiltonians for quantum simulation. PRX Quantum, 2021, 2: 020329
26 McArdle S, Endo S, Aspuru-Guzik A, et al. Quantum computational chemistry. Rev Mod Phys, 2020, 92: 015003
27 Aspuru-Guzik A, Dutoi A D, Love P J, et al. Simulated quantum computation of molecular energies. Science, 2005, 309: 1704–1707
28 Wei S J, Chen Y H, Zhou Z R, et al. A quantum convolutional neural network on NISQ devices. AAPPS Bull, 2022, 32: 2
29 Huang H L, Du Y, Gong M, et al. Experimental quantum generative adversarial networks for image generation. Phys Rev Appl, 2021, 16: 024051
30 Beer K, Bondarenko D, Farrelly T, et al. Training deep quantum neural networks. Nat Commun, 2020, 11: 808
31 Rebentrost P, Gupt B, Bromley T R. Quantum computational finance: Monte Carlo pricing of financial derivatives. Phys Rev A, 2018, 98: 022321
32 Tang H, Pal A, Wang T Y, et al. Quantum computation for pricing the collateralized debt obligations. Quantum Eng, 2021, 3: 84
33 Egger D J, Gambella C, Marecek J, et al. Quantum computing for finance: state-of-the-art and future prospects. IEEE Trans Quantum Eng, 2020, 1: 1–24
34 Tuchman W. A brief history of the data encryption standard. In: Internet Besieged: Countering Cyberspace Scofflaws. New York: ACM Press/Addison-Wesley Publishing Co., 1997. 275–280
35 Nelder J A, Mead R. A simplex method for function minimization. Comput J, 1965, 7: 308–313

Appendix A The details of S-DES

The generation of sub-key. Firstly, the initial key can be arranged as $(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$. The sub-keys K_1, K_2 can be generated as

$$\begin{aligned} K_1 &= \mathrm{P8}[\mathrm{Shift}_1[\mathrm{P10}[\mathrm{key}]]], \\ K_2 &= \mathrm{P8}[\mathrm{Shift}_2[\mathrm{Shift}_1[\mathrm{P10}[\mathrm{key}]]]]. \end{aligned} \tag{A1}$$

The replacement function P10 is defined as

$$\mathrm{P10}\,(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)\,. \tag{A2}$$

So the first bit of the output is the third bit of the input, the second bit of the output is the fifth bit of the input, etc. Then the first 5 bits and the last 5 bits are shifted to the left by one bit (shift_1), respectively. Next, we select 8 bits from the above 10 bits by P8 as the sub-key K_1. where P8 is expressed as

P8							
6	3	7	4	8	5	10	9

(A3)

In Eq. (A3), the numbers represent the positions in the 10-bit string of Eq. (A2) and the figures in this section have the same meaning.

Finally, let us return to the 5-bit string pairs produced by the function Shift_1, then shift them to the left by two bits (Shift_2). Turning to the sub-key K_2 now, it is obtained by the operation P8 of Eq. (A3).

The encryption. For an 8-bit plaintext, the first equation is IP:

IP							
2	6	3	1	4	8	5	7

(A4)

This function retains the 8-bit information of the plaintext but rearranges it. The inversion function of IP is

IP^{-1}							
4	1	3	5	7	2	8	6

(A5)

The most complex part of S-DES is the function f_K, which consists of the replacement and substitution operations. To be specific, f_K is expressed as

$$f_K(L,R) = (L \oplus F(R,K), R), \tag{A6}$$

where L and R are the left 4 bits and right 4 bits of the 8-bit input, F is a mapping from 4 bits to 4 bits, K is the sub-key, $\oplus$ is XOR operation. Assuming that the output of the function IP is 10111101, for some key K, we have $F(1101, K) = 1110$. Because we have $1011 \oplus 1110 = 0101$, this yields $f_K(10111101) = 01011101$.

Now we illustrate the map F. Its input is a 4-bit string: (n_1, n_2, n_3, n_4). The first operation is its extension

E/P							
4	1	2	3	2	3	4	1

(A7)

which can be expressed as

$$\begin{array}{c|cc|c} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{array}. \tag{A8}$$

Upon performing XOR with the 8-bit sub-key $K1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$, we arrive at

$$\begin{array}{c|cc|c} n_4 \oplus k_{11} & n_1 \oplus k_{12} & n_2 \oplus k_{13} & n_3 \oplus k_{14} \\ n_2 \oplus k_{15} & n_3 \oplus k_{16} & n_4 \oplus k_{17} & n_1 \oplus k_{18} \end{array}. \tag{A9}$$

It can be recorded as

$$\begin{array}{c|cc|c} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \end{array}. \tag{A10}$$

The first 4 bits (the first row in Eq. (A10)) and the last 4 bits (the second row in Eq. (A10)) are imported into the S-box S_0 and S-box S_1 and the 2-bit output strings, respectively. The pair of S-boxes are as follows:

$$S_0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \end{array}, \quad S_1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix} \end{array}. \tag{A11}$$

The first and forth bits of the 4-bit input form a 2-bit binary number, which represents the S-box row, while the second and third bits represent the S-box column. The element determined by the row and column of S-box is the output, which is a 2-bit binary number.

Then the 4-bit output from S_0 and S_1 is ordered by the operation P4:

P4			
2	4	3	1

(A12)

The output of P4 is the result of the function F.

The function f_K just changes the 4 bits on the left, while the function SW is an exchange function, which swaps the left 4 bits and right 4 bits of the input, so the input of function f_K in the second round is represented by four different bits.

Appendix B The performance of different regular graphs

In Table B1, we show the average number of iterations for different regular graphs where the initial parameters are the same and the other parameters have been set as the proper values. The data is calculated by 15 simulations with the Y-Cz (A) ansatz and the gradient method.

Table B1 The number of iterations for different regular graphs

	1-reg	2-reg	3-reg	4-reg	5-reg	6-reg	7-reg
Learning rate	0.72	0.84	1.08	1.44	1.92	2.40	2.88
Restart condition	0.53	0.62	0.80	1.07	1.42	1.78	2.13
Maximum	94	94	94	94	94	94	94
Minimum	4	2	4	2	2	1	14
Average	53.27	33.07	31.13	32.73	34.87	44.93	65.60

When the norm of gradient is less than the value of Restart condition, the optimization will be restarted. From the results of Table B1, we find the 3-regular graph performs best.

Appendix C Two classical optimization algorithms

In Algorithm C1, the learning rates in the A-ansatz are set to 0.72, 0.72, 1.08 for the Y-Cx model, Y-Cy model, Y-Cz model respectively; the learning rates in the B-ansatz are set to 0.72, 0.76, 0.94 for the Y-Cx model, Y-Cy model, Y-Cz model, respectively.

Algorithm C1 Gradient descent method

```
Require: Initial point $x_0$, the function $f$, the learning rate $r$, the cut-off condition xerr.
 1: times = 0;
 2: Calculate len = length($x_0$);
 3: for $ii$ in range(1024) do
 4:     Let cost = $f(x_0)$;
 5:     times = times + 1;
 6:     if costf < xerr then
 7:         Break;
 8:     end if
 9:     Initialize a zero vector Gd, the length is len;
10:     for $i$ in range(len) do
11:         Let $x \leftarrow x_0$;
12:         Change the $i$-th component of $x$: $x_i = x_i + 0.01$;
13:         $\mathrm{cost}' = f(x)$;
14:         times = times + 1;
15:         The $i$-th component of Gd: $\mathrm{Gd}_i = (\mathrm{cost}' - \mathrm{cost})/0.01$;
16:     end for
17:     Generate a random number $r_0$ in range $[0, 1]$;
18:     $x_0 = x_0 - (\mathrm{r}/|\mathrm{cost}| + \log(\mathrm{times})/\mathrm{times} \times r_0) \times \mathrm{Gd}$;
19:     if $|\mathrm{Gd}| < 0.8$ then
20:         Initialize a $x_0$ randomly;
21:     end if
22: end for
23: return $x$.
```

Algorithm C2 N-M method

```
Require: Generating the other $N$ (the dimension of $x$) points $(x_1, \ldots, x_N)$ according to the initial point $x_0$. Let the $i$-th component
    in $x_i$ is $\alpha$ (the amplification factor) larger than the $i$-th component in $x_0$. If the $i$-th component is 0 in $x_0$, the $i$-th component
    in $x_i$ is set to 0.8, the cut-off condition xerr.
 1: times $= N + 1$;
 2: while times $< 1024$ do
 3:     Sort and rename these points $(x_i)$ in ascending order according to the value of $f(x_i)$, the larger $i$, the larger $f(x_i)$;
 4:     if $f(x_0) \leqslant$ xerr then
 5:         Break;
 6:     end if
 7:     if $f(x_N) - f(x_0) < 0.15$ then
 8:         Restart;
 9:     end if
10:     Calculate the average of the first $N$ points $m = \frac{1}{N}\sum_{i=0}^{N-1} x_i$;
11:     Calculate the reflect point $r = 2m - x_N$;
12:     times = times + 1;
13:     if $f(x_1) \leqslant f(r) < f(x_{N-1})$ then
14:         $x_N = r$; Continue;
15:     end if
16:     if $f(r) < f(x_1)$ then
17:         Calculate the expand point $s = m + 2(m - x_N)$;
18:         times = times + 1;
19:         if $f(s) < f(r)$ then
20:             $x_N = s$; Continue;
21:         else
22:             $x_N = r$; Continue;
23:         end if
24:     end if
25:     if $f(x_{N-1}) \leqslant f(r) < f(x_N)$ then
26:         $c_1 = m + (r - m)/2$;
27:         times = times + 1;
28:         if $f(c_1) < f(r)$ then
29:             $x_N = c_1$; Continue;
30:         else
31:             $v_i = x_0 + (x_i - x_0)/2$; $x_i = v_i$ $(i = 1, \ldots, N)$;
32:             times = times + $N$;
33:             Continue;
34:         end if
35:     end if
36:     if $f(x_N) \leqslant f(r)$ then
37:         $c_2 = m + (x_N - m)/2$;
38:         times = times + 1;
39:         if $f(c_2) < f(x_N)$ then
40:             $x_N = c_2$; Continue;
41:         else
42:             $v_i = x_0 + (x_i - x_0)/2$; $x_i = v_i$ $(i = 1, \ldots, N)$; times = times + $N$;
43:             Continue;
44:         end if
45:     end if
46: end while
47: return $x_0$.
```

In Algorithm C2, the amplification factors in the A-ansatz are set to 2.7, 2.7, 2.8 for the Y-Cx model, Y-Cy model, Y-Cz model, respectively; the amplification factors in the B-ansatz are set to 2.7, 2.7, 2.7 for the Y-Cx model, Y-Cy model, Y-Cz model, respectively.