Attacking classical cryptosystems with quantum adversaries that leverage Simon's Algorithm

Karan Grewal, Xing Hu, Zhewei Sun

Abstract

This report examines how Simon's algorithm can be used to break classical cryptosystems in the presence of a quantum adversary. In particular, we show how to use Simon's algorithm to distinguish between a Feistel cipher and a random permutation, which is provably impossible using classical queries only. We also show how to use Simon's algorithm to break the security of classical message authentication codes by forging message that were never queried by the quantum attacker. Besides, we study in depth how unwanted collisions in cryptographic oracles may affect the efficiency of Simon's algorithm.

1 Introduction

Cryptography concerns secure communication between two parties in the presence of a third party adversary. Many popular cryptographic systems use encryption methods in which the sender and receiver both use the same key, known as *symmetric-key cryptography*. Although these cryptographic systems have been proven secure against classical adversaries (i.e., whose operations can be executed on a standard Turing machine), recent advancements in the field of quantum computation have demonstrated how these communication channels are susceptible to adversaries that leverage quantum computers. These quantum adversaries leverage Simon's Algorithm [1] to break classical cryptographic system. As many researchers are now pursuing the advent of quantum computers for ubiquitous use, classical cryptographic systems are in at risk due to their inability to tolerate a quantum adversary. In subsequent sections, we analyze theoretical properties of Simon's Algorithm and discuss how can be applied to break symmetric-key cryptographic systems, such as 3-round Feistel networks and message authentication codes.

2 Analyzing Simon's Algorithm

Simon's algorithm [1] is one of the simplest quantum algorithms for period finding first proposed by Daniel Simon in 1994. Although the original computational problem proposed by Simon has little practical value, the algorithm can be applied to various problems with appropriate tweaks. Here, we present the findings of [2] on how Simon's algorithm can be modified to serve as an integral component in quantum attacks. In particular, Simon's algorithm assumes to operate on functions with a very specific collision structure but it has been shown that such assumptions can be neglected by increasing the number of iterations by a constant factor. In this section, we first present the original Simon's algorithm in a way that best illustrates the necessity of the assumption. We then showcase potential issues with having unwanted collisions as well as results that guarantee a lower bound on the success rate of Simon's algorithm in such scenarios. Finally, we present some high level ideas used to prove such bounds.

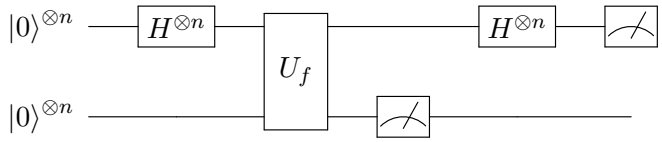
2.1 Simon's problem and alternative formulation

Simon's problem is a phase finding problem that is difficult to solve in a classical setting but can be efficiently solved by quantum computers. The problem can be formulated as follows:

Simon's Problem: Given a function $f : \{0,1\}^n \to \{0,1\}^n$ such that there exists a hidden phase $s \in \{0,1\}^n$ so that $f(x) = f(y)$ iff $x \oplus y \in \{0^n, s\}$ for all $x, y \in \{0,1\}^n$. Find the hidden phase s.

This formulation of the problem is slightly different than the one presented in class. Instead of having $x \oplus s = y$, we have $x \oplus y = s$. But since the addition and subtraction operators are equivalent in $\mathbb{Z}$ mod 2, the two expressions are equivalent and so are the formulations.

To solve Simon's problem, we run the following quantum circuit representing the Simon's algorithm:



where $H^{\otimes n}$ refers to applying the Hadamard gate to n qubits in parallel. Recall from class that the n-bit Hadamard unitary can be expressed as:

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \tag{1}$$

The dot product $x \cdot y = x_1 y_1 \oplus \ldots \oplus x_n y_n$ essentially computes the parity of the pairwise multiplied elements $x_i y_i$'s.

We now analyze the circuit step by step[1]:

1. The first n Hadamard gates creates a uniform super position over the first set of qubits:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle$$

2. Applying the oracle unitary gives:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

3. Now we measure the latter n qubits in our circuit. Since these qubits are in a uniform superposition over the image set of f, measuring these qubits will result in a value $f(z)$ for some $z \in Im(f)$ with uniformly distributed probability. Our function f has the property that $f(x) = f(y)$ iff $x \oplus y \in \{0^n, s\}$, thus the pre-image of $f(z)$ is exactly $x = z$ and $x = z \oplus s$. (i.e. $f(x) \neq f(z)$ for all other x's). Therefore, the post-measurement state on the first n qubits consist of exactly these two super-positions:

$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle)$$

[1] We write $|0\rangle$ to represent $|0\rangle^{\otimes n}$ for ease of notation.

4. Applying Hadamard to the post-measurement state on the first n qubits gives:

$$\begin{aligned}
&\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}|y\rangle + (-1)^{y\cdot(z\oplus s)}|y\rangle \\
=&\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}|y\rangle + (-1)^{y\cdot z}(-1)^{y\cdot s}|y\rangle \\
=&\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}(1+(-1)^{y\cdot s})|y\rangle
\end{aligned}$$

where we have used the alternate formulation of $H^{\otimes n}$ as in eq. (1).

Notice the term $(1+(-1)^{y\cdot s})$ vanishes iff $y\cdot s = 1$. Thus measuring the qubits would uniformly randomly select a vector $s \in \{0,1\}^n$ such that $y \cdot s = 0$ (i.e. y is orthogonal to s).

We can then repeat the above subroutine $O(n)$ times to obtain $n-1$ independent orthogonal vectors. This would allow us to uniquely determine s since we're working with unit length vectors in an n-dimensional vector space.

2.2 Imperfect promise due to unwanted collisions

In the above formulation of Simon's Problem, we have assumed perfect collision in the function f, meaning that collisions would only occur with phase s (i.e. $f(x) = f(y) \Rightarrow x \oplus y = s$)[2]. This is not a reasonable assumption to make for functions in general because collisions could happen spuriously.

However, this may jeopardize the practicality of Simon's algorithm as step 3 of our analysis above relied on f having perfect collisions. Additional states $|x\rangle$ with $f(x) = f(z)$ will be introduced after measurement in step 3:

$$\frac{1}{\sqrt{|f^{-1}(z)|}}(|z\rangle + |z\oplus s\rangle + \sum_{x\in A_z}|x\rangle)$$

where $A_z = f^{-1}(z)\setminus\{z, z\oplus s\}$ contains elements where the spurious collisions happen. Consequently, the qubit states after step 4 also changes accordingly:

$$\frac{1}{\sqrt{|f^{-1}(z)|}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}|y\rangle + (-1)^{y\cdot(z\oplus s)}|y\rangle + \sum_{x\in A_z}(-1)^{y\cdot x}|y\rangle$$

In this case, unless y is orthogonal to s, the first two terms would still cancel out but the additional terms remain relevant. Luckily, elements in A_z are also symmetric due to the hidden phase. Specifically, A_z can be partitioned evenly into two sets A'_z, A''_z such that if $x \in A'_z$, then $x\oplus s \in A''_z$. Then, $\{z\}\cup A'_z$ will include exactly half of $f^{-1}(z)$ and our quantum state then becomes:

$$\begin{aligned}
&\frac{1}{\sqrt{|f^{-1}(z)|}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}|y\rangle + (-1)^{y\cdot(z\oplus s)}|y\rangle + \sum_{x\in A'_z}(-1)^{y\cdot x}|y\rangle + (-1)^{y\cdot(x\oplus s)}|y\rangle \\
=&\frac{1}{\sqrt{|f^{-1}(z)|}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{y\cdot z}(1+(-1)^{y\cdot s})|y\rangle + \sum_{x\in A'_z}(-1)^{y\cdot x}(1+(-1)^{y\cdot s})|y\rangle \\
=&\frac{1}{\sqrt{|f^{-1}(z)|}}\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}\sum_{x\in\{z\}\cup A'_z}(-1)^{y\cdot x}(1+(-1)^{y\cdot s})|y\rangle
\end{aligned}$$

[2]In class, this assumption was made implicitly by saying that the size of $Im(f)$ is exactly $\frac{1}{2}$ of the domain.

Therefore, we still have a zero measurement probability for any vector y that is not orthogonal to s (i.e. $y \cdot s = 1$). But for the orthogonal vectors, the measurement probabilities are no longer uniformly distributed. Instead, the probability of measuring y becomes:

$$\begin{aligned} Pr(y) &= \sum_{f(z)\in Im(f)} Pr(f(z)) \cdot Pr(y|f(z)) \\ &= \sum_{f(z)\in Im(f)} \frac{|f^{-1}(z)|}{2^n} \cdot \left\| \frac{1}{\sqrt{|f^{-1}(z)|}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{z\}\cup A'_z} (-1)^{y\cdot x}(1+(-1)^{y\cdot s})|y\rangle \right\|^2 \\ &= \sum_{f(z)\in Im(f)} \frac{4}{2^{2n}} \left(\sum_{x\in\{z\}\cup A'_z} (-1)^{y\cdot x} \right)^2 \end{aligned} \tag{2}$$

Ideally, we want the distribution over y to be uniform. The uniformity of y's is important because Simon's algorithm needs to efficiently extract a set of linearly independent vectors orthogonal to s. A distribution skewed towards a subset of the dimensions may undermine our ability to do this efficiently.

In the case without spurious collisions, we have $|Im(f)| = \frac{1}{2}$ and $A'_z = \emptyset$, thus the distribution of y's are uniform. Otherwise, the probability of measuring y depends on the distribution of $y \cdot x$ over elements x's in the pre-image of z. Ideally, we want $y \cdot x$ to be closely distributed between 0 and 1 so that none of the vectors y's will have high measurement probability.

2.3 Probabilistic bounds on success rate

The analysis above suggests that Simon's algorithm may not achieve successful results efficiently. KLLNP [2] shows that by bounding the number of spurious collisions in f, it is possible to bound the number of required iterations to be polynomial in n. We quantify collision frequency of f as follows:

$$\epsilon(f,s) = \max_{t\in\{0,1\}^n\setminus\{0,s\}} Pr_x[f(x) = f(x\oplus t)]$$

Theorem 1 [KLLNP16] if $\epsilon(f,s) \le p_0 < 1$, then Simon's algorithm returns s with cn queries, with probability at least $1 - (2(\frac{1+p_0}{2})^c)^n$. In particular, choosing $c \ge \frac{3}{1-p_0}$ ensures the error probability decreases exponentially with n.

The above theorem says that the existence of unwanted collisions only brings in a polynomial increase in the number of iterations, given that $\epsilon(f,s)$ is bounded away from 1. However, this is not always true in practice as there may exist multiple phases t such that $f(x) = f(x\oplus t)$ for all x. In other words, $Pr_x[f(x) = f(x\oplus t)] = 1$ for some t and thus $\epsilon(f,s) = 1$. Although we can no longer ambiguously retrieve s in this case, it is possible to show that any vectors orthogonal to the set of vectors returned by Simon's algorithm is a phase of f with high probability:

Theorem 2 [KLLNP16] After cn steps of Simon's algorithm, if t is orthogonal to all vectors u_i returned by each step of Simon's algorithm, then $Pr_x[f(x) = f(x\oplus t)] \ge p_0$ with probability at least $1 - (2(\frac{1+p_0}{2})^c)^n$. In particular, choosing $c \ge \frac{3}{1-p_0}$ ensures that the probability is exponentially close to 1.

For example, setting $p_0 = 0.99$ would ensure success 99% of the time with 300 times the number of iterations. Therefore, in either case the algorithm will return the desired result with high probability with $c = \frac{3}{1-p_0}$ times the iterations, despite the existence of unwanted collisions.

We provide an outline of the proofs in appendix B.

3 General attack strategy based on Simon's algorithm

The general strategy behind these attacks based on Simon's algorithm is to start with the encryption oracle $E_k : \{0,1\}^n \to \{0,1\}^n$ and find a new function f, call it "Simon's function", which satisfies special properties such that the period information of f can be found by Simon's algorithm and it's sufficient to break the cryptographic scheme.

3.1 Simon's function

The Simon's function f is handcrafted from the encryption scheme E_k to be attacked and it satisfies the following three properties:

1. $\exists s,\ f(x) = f(x \oplus s)$
2. Can be queried in superposition if an adversary has quantum oracle access to E_k.
3. The information of the string s is sufficient to break the cryptographic scheme.

3.2 Attack outline

Given encryption scheme E_k:

1. find Simon's function f
2. use Simon's algorithm to find s such that $f(x) = f(x \oplus s)$
3. use s to break E_k

4 Quantum attacker of 3-round Feistel Cipher

Some cryptographic concepts used in the following sections are provides in appendix. A

4.1 3-round Feistel construction

Let F be a pseudorandom function $\{0,1\}^n \to \{0,1\}^n$. Let $k = k_1||k_2||k_3$ be a random key with length $3n$. Let V be a 3-round Feistel scheme with internal pseudorandom function F and key k. Denote V's encryption Enc_v. Given $x \in \{0,1\}^{2n}, x = a||c$, where $a, c \in \{0,1\}^n$ and $||$ is the concatenation operator on strings, Figure 1 illustrates 3-round Feistel construction. The encryption of x using V is as follows:

$$Enc_v(x) = Enc_v(a||c) = c \oplus F_{k_2}(a \oplus F_{k_1}(c))||(a \oplus F_{k_1}(c)) \oplus F_{k_3}(c \oplus F_{k_2}(a \oplus F_{k_1}(c)))$$

We recall the following theorem from [3], which proved the security of 3-round Feistel against classic adversary :

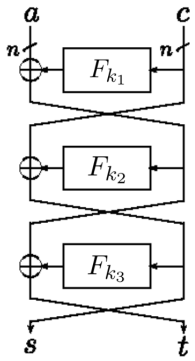


Figure 1: 3-Round Feistel Construction

Theorem 3 [LR88] *If F is a pseudorandom function, then the 3-round Feistel construction with internal round functions $F_{k_1}, F_{k_2}, F_{k_3}$ is a pseudorandom permutation mapping 2n-bit strings to 2n-bit strings,and using a key $k = k_1||k_2||k_3$ of length $3n$.*

Therefore, a classical adversary needs exponential number of queries to distinguish between Feistel Cipher with 3 (or more) rounds and a random permutation.

4.2 Attack 3-round Feistel with internal pseudorandom permutations

In this section, we consider an attack to 3-round Feistel schemes where $F_{k_1}, F_{k_2}, F_{k_2}$ are pseudorandom **permutations**. This problem is first studied in [4] and then further studied in [2, 5]. The following survey is mainly based on [4].

Let α, β be two distinct fixed strings in $\{0,1\}^n$. Consider the Simon's function $f : \{0,1\}^{n+1} \to \{0,1\}^n$

$$f(b||a) = \begin{cases} \alpha \oplus P_2(a \oplus P_1(\alpha)) \oplus \beta, & b = 0 \\ \beta \oplus P_2(a \oplus P_1(\beta)) \oplus \alpha, & b = 1 \end{cases}$$

where $b \in \{0,1\}, a \in \{0,1\}^n$.

It is easy to verify that $f(x) = f(x \oplus (1||P_1(\alpha) \oplus P_1(\beta)))$, then $s = 1||P_1(\alpha) \oplus P_1(\beta)$ is the period needed to break pseudorandomness of the 3-round Feistel scheme.

If $x = 0||a$,

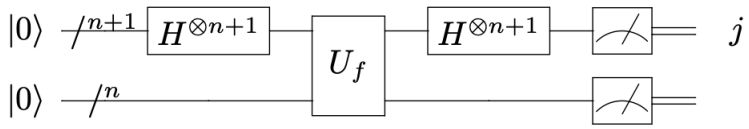
$$\begin{aligned} f(x \oplus (1||P_1(\alpha) \oplus P_1(\beta))) &= f(1||a \oplus P_1(\alpha) \oplus P_1(\beta)) \\ &= P_2(a \oplus P_1(\alpha) \oplus P_1(\beta) \oplus P_1(\beta)) \\ &= P_2(a \oplus P_1(\alpha)) \\ &= f(0||a) \end{aligned}$$

If V is 3-round Feistel, $f(u) = f(u')$.

If V is truly random, s is random and $f(u) = f(u \oplus s)$ happens only with negligible probability.

The attack based on Simon's algorithm is as follows:

1. Initialize $J = \{\}$
2. Run Simon's subroutine where f is the function defined above:



3. Add the measurement result j to J. If J does not contain n lineally independent j's, then go back to step 2. Otherwise, find an $n+1$-bit string s where $s_0 = 1$ by solving the linear system of equations:

$$\begin{pmatrix} | & & | \\ j_1 & \cdots & j_n \\ | & & | \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} | \\ j_0 \\ | \end{pmatrix}$$

4. Choose an $(n+1)$-bit string u at random. Let $u' = u \oplus s$. Compute $f(u)$ and $f(u')$ classically. If they are equal, then V is guessed to be the 3-round feistel cipher. Otherwise V is guessed to be random.

Using the same analysis as of Simone's algorithm, the period string s will be found using $O(n)$ queries in expectation.

4.3 Attack 3-round Feistel with internal pseudorandom function

In this section, we consider an attack to 3-round Feistel scheme where $F_{k_1}, F_{k_2}, F_{k_2}$ are pseudorandom **functions**. The generation from internal pseudorandom permutation to pseudorandom function is studied in [2, 5].

When generalizing to 3-round Feistel scheme with internal pseudorandom functions, there might be multiple s's such that $f(x) = f(x \oplus s)$ for the function f defined above since the internal pseudorandom functions $F_{k_1}, F_{k_2}, F_{k_3}$ may not be bijective.[3] Therefore, f doesn't satisfy the requirement of Simon's algorithm which requires unique s. This issue increases the probability of not finding n linearly independent vectors using $O(n)$ queries.

[5] and [2] both addressed this issue. In [5], after each subroutine, the attacker also checks if there are already $2n$ vectors in J but no n linearly independent vectors. If so, stop and guess V to be 3-round Feistel construction. Provably, if V is a random function, the probability that V has no n linearly independent vector after $2n$ round is negligible. In [2], they proved that $\epsilon(f, 1||s) < \frac{1}{2}$

[3] With internal permutations, $s \in \{0, 1||p_1(\alpha) \oplus p_1(\beta)\}$ are the only strings such that $f(x) = f(x \oplus s)$.

with overwhelming probability if V is a random function.

These two solutions have similar ideas that while generating from permutation to function may bring unwanted collisions, the probability that Simon's algorithm is influenced by these collisions is negligible.

5 Applications to Message Authentication Codes

In this section, we present applications of Simon's algorithm to forge messages. A message authentication code (MAC) is used to guarantee the authenticity of a message. If Alice wants to send a message m to Bob, she feeds a key k and message m to an algorithm that computes $t = \mathsf{MAC}_k(m)$. Bob then receives (m, t) and can verify if t is the correct tag for message m by using the same key k to query the MAC algorithm.

5.1 Forging tags of fixed-sized messages

A cipher block chaining message authentication code (CBC-MAC) is one of the earliest implementations of a MAC. CBC-MACs use block ciphers, deterministic algorithms that act on an n-bit block, as an encoding scheme. If Alice wants to send the message $m = m_1||\dots||m_\ell$ to Bob where each m_i consists of n-bits and there are ℓ such blocks, the CBC-MAC encoding scheme is

$$x_0 = 0^n, \qquad x_i = E_k(x_{i-1} \oplus m_i), \qquad \mathsf{CBCMAC}_k(m) = E_k(x_\ell)$$

where each E_k is a block cipher that is determined by the secret key k agreed upon by Alice and Bob[4]. This way, Alice computes $t = \mathsf{CBCMAC}_k(m)$ and sends (m, t) to Bob who then verifies if t is the correct tag for the message given k. Figure 2 illustrates this encoding process.

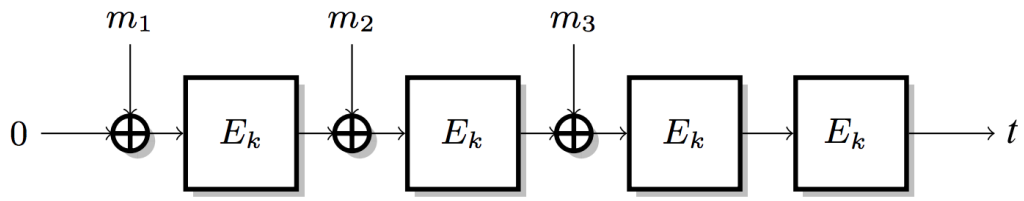


Figure 2: The CBC-MAC encoding scheme for a message $m = m_1||m_2||m_3$ that has three blocks.

We now show how to break the CBC-MAC encryption scheme, as first shown in [2] and [5]. Here we assume the quantum adversary has access to the oracle E_k and consider only n-bit messages m preceeded by a prefix α_0. Fix another n-bit block α_1 and define the function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ by

$$b, x \mapsto \mathsf{CBCMAC}(\alpha_b||x) := E_k(E_k(x \oplus E_k(\alpha_b))).$$

Notice that $f(b, x) = f(b', x')$ iff $x \oplus E_k(\alpha_b) = x' \oplus E_k(\alpha_{b'})$. There are 2 cases:

- **Case a.** If $b = b'$, then $x \oplus x' = 0^n$ which implies $x = x'$.

[4]Note that here $x_0 = 0^n$ denotes a classical n-bit string.

- **Case b.** Otherwise if $b \neq b'$, then WLOG, $x \oplus E_k(\alpha_0) = x' \oplus E_k(\alpha_1)$ and since we are working with addition mod 2, $x \oplus x' = E_k(\alpha_0) \oplus E_k(\alpha_1)$.

The function f thus satisfies for all $b, b' \in \{0,1\}$ and for all $x, x' \in \{0,1\}^n$, $f(b,x) = f(b',x')$ iff $x = x' \oplus E_k(\alpha_0) \oplus E_k(\alpha_1)$. By applying Simon's algorithm, the n bits $s = E_k(\alpha_0) \oplus E_k(\alpha_1)$ are retrieved despite not knowing the prefix α_0.

Now the quantum adversary wants to forge m and send it to Bob. For this, it needs to compute the tag $t = \mathsf{CBCMAC}(\alpha_0||m) := E_k(E_k(\alpha_0) \oplus m)$. Thanks to Simon's Algorithm, the quantum adversary can generate this tag by querying $\alpha_1||m \oplus s$ as follows:

$$\begin{aligned}\mathsf{CBCMAC}(\alpha_1||m \oplus s) &= \mathsf{CBCMAC}(\alpha_1||m \oplus E_k(\alpha_0) \oplus E_k(\alpha_1)) \\ &= E_k(E_k(\alpha_1) \oplus m \oplus E_k(\alpha_0) \oplus E_k(\alpha_1)) \\ &= E_k(E_k(\alpha_0) \oplus m) \\ &= t.\end{aligned}$$

The quantum adversary has thus generated the correct tag for the message $\alpha_0 \parallel m$ without knowing the prefix.

5.2 Forging tags of non-queried messages

The typical notion of secure encryption scheme states that it is quantum secure if there is no quantum adversary that make only q queries yet still produces $q+1$ valid tags. Instead of invalidating this definition of security, we will show that the CBC-MAC encryption scheme is not quantum secure because a quantum adversary can forge the tag of a message without querying it. Let $m = m_1 \parallel \ldots \parallel m_\ell$ be a message of variable size which has length ℓn bits. Define $f_j : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ as

$$b \parallel x \mapsto \mathsf{CBCMAC}_k(\alpha_b \parallel 0^{(j-1)n}) := E_k(E_k^j(\alpha_b) \oplus x)$$

where E_k^j denotes j applications of the oracle E_k. The function f_j essentially computes the tag of a message in which all but the j^{th} block are necessarily 0^n. Similar to result in the previous section, f_j satisfies $f(b,x) = f(b',x')$ iff $x = x'$ or $x = x' \oplus E_k^j(\alpha_0) \oplus E_k^j(\alpha_1)$. Thus a quantum adversary can apply Simon's algorithm to obtain $s_j = E_k^j(\alpha_0) \oplus E_k^j(\alpha_1)$.

The quantum adversary can produce a valid tag for the message $m^* = \alpha_1 \parallel s_1 \parallel \ldots \parallel s_{\ell-1}$ as follows:

1. Apply Simon's algorithm a total of $\ell - 1$ times (to each f_j where the query uses input m_j) to obtain s_j for $j = 1, \ldots, \ell - 1$.
2. Query the messages comprising all zeros $0^{(\ell-1)n}$ to get two tags $t_0 = \mathsf{CBCMAC}_k(\alpha_0 \parallel 0^{(\ell-1)n}) = E_k^\ell(\alpha_0)$ and $t_1 = \mathsf{CBCMAC}_k(\alpha_1 \parallel 0^{(\ell-1)n}) = E_k^\ell(\alpha_1)$.
3. Forge the desired tag t as $t = t_0$ if ℓ is even, and $t = t_1$ is ℓ is odd.

Now we show that t is the correct tag for m^* by induction.

Base Case. If $\ell = 2$, then

$$\begin{aligned}\mathsf{CBCMAC}_k(\alpha_1 \parallel s_1) &= E_k(E_k(\alpha_1) \oplus s_1) \\ &= E_k(E_k(\alpha_1) \oplus E_k(\alpha_0) \oplus E_k(\alpha_1))) \\ &= E_k^2(\alpha_0) \\ &= t_0.\end{aligned}$$

Otherwise if $\ell = 3$, then

$$\begin{aligned}
\mathsf{CBCMAC}_k(\alpha_1 \parallel s_1 \parallel s_2) &= E_k(E_k(E_k(\alpha_1) \oplus s_1) \oplus s_2) \\
&= E_k(E_k(E_k(\alpha_1) \oplus E_k(\alpha_0) \oplus E_k(\alpha_1))) \oplus s_2) \\
&= E_k(E_k^2(\alpha_0) \oplus E_k^2(\alpha_0) \oplus E_k^2(\alpha_1)) \\
&= E_k^3(\alpha_1) \\
&= t_1.
\end{aligned}$$

Hypothesis. Assume $\mathsf{CBCMAC}_k(\alpha_1 \parallel s_1 \parallel \ldots \parallel s_{\ell-1}) = E_k^\ell(\alpha_b)$ where $b = 0$ if ℓ is even and 1 if ℓ is odd.

Induction Step. If ℓ is odd and $\ell + 1$ is even, then

$$\begin{aligned}
\mathsf{CBCMAC}_k(\alpha_1 \parallel s_1 \parallel \ldots \parallel s_\ell) &= E_k(E_k(\ldots E_k(E_k(\alpha_1) \oplus s_1) \ldots) \oplus s_{\ell-1}) \oplus s_\ell) \\
&= E_k(E_k^\ell(\alpha_1) \oplus s_\ell) \\
&= E_k(E_k^\ell(\alpha_1) \oplus E_k^\ell(\alpha_0) \oplus E_k^\ell(\alpha_1)) \\
&= E_k^{\ell+1}(\alpha_0) \\
&= t_0.
\end{aligned}$$

A similar induction step proves that $\mathsf{CBCMAC}_k(m^*) = t_1$ if ℓ is even and $\ell + 1$ is odd. In summary, a quantum adversary can therefore forge the tag of a message $\alpha_1 \parallel s_1 \parallel \ldots \parallel s_{\ell-1}$ without actually querying it. Simon's Algorithm can also be used to break others MACs, such as parallelizable MACs, where an encryption code is computed for each message block in parallel, and randomized MACs, where the block cipher is a hash function [2].

6 Conclusion

We have shown that some classical cryptographic systems are easily susceptible to attacks by quantum adversaries. These adversaries could leverage Simon's Algorithm to devise efficient quantum attacks that jeopardize the security of 3-round Feistel networks and various types of message authentication codes. As it stands, not all symmetric key cryptographic systems are ready for the post-quantum world and future work is needed to ensure privacy and protection against malicious parties with access to quantum computational resources.

References

[1] D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

[2] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems usingquantumperiod finding. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 207–237, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[3] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[4] H. Kuwakado and M. Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, June 2010.

[5] Thomas Santoli and Christian Schaffner. Using simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Info. Comput.*, 17(1-2):65–78, February 2017.

Appendix A Basic Cryptographic concepts

The typical security proof is to prove that a cryptographic scheme is indistinguishable from an ideal scheme, which is random. The following cryptographic concepts are frequently used in the survey.

Definition A.1. Cipher/Encryption scheme [Informal]
A cipher/encryption scheme is a pair of algorithms that create the *encryption*, which is the process of converting ordinary information (called plaintext) into unintelligible form (called ciphertext) and the reversing *decryption*, which is moving from the unintelligible ciphertext back to plaintext.

Definition A.2. Pseudorandom Function Family [Informal]
A pseudorandom function family F, is a collection of efficiently-computable functions which emulate a random oracle in the following way: no efficient algorithm can distinguish (with significant advantage) between a function chosen by random key k from F and a random oracle (a function whose outputs are fixed completely at random).

Definition A.3. Pseudorandom Permutation[5] Family [Informal]
A pseudorandom permutation family F, is a collection of efficiently-computable permutations which cannot be distinguished from truly random permutations by any efficient procedure that can get the values of the permutations at arguments of its choice.

Appendix B Outline for proofs in Section 2.3

In this section, we present some high level ideas used to prove the results outlined in section 2.3. In the interest of space, we omit the full proofs given in the Appendix of KLLNP16.

1. Let $t \in \{0,1\}^n$ and let $t^\perp$ be the set of all vectors orthogonal to t (i.e. $t^\perp = \{y \in \{0,1\}^n \text{ s.t. } y \cdot t = 0\}$. Assuming $t \neq 0$, then $|t^\perp| = \frac{1}{2}|\{0,1\}^n| = 2^{n-1}$.

   Since t is non-zero, we can view this as solving a linear system with 1 equation, thus reducing the dimensionality by 1. Thus the set $t \perp$ has 2^{n-1} elements in total.

2. Let $x \in \{0,1\}^n$ and let $E_i = \{y \in t^\perp \text{ s.t. } y \cdot x = i\}$. Then $|E_0| = |t^\perp|$ if $x = 0$ or $x = t$, otherwise, $|E_0| = |t^\perp|/2$

   Similar to the previous case, except we now have two equations to constrain our space: $y \cdot t = 0$ and $y \cdot x = 0$. Since x is non-zero and $x \neq t$, the two equations are linearly independent, therefore resulting in a subspace of dimension $n-2$ with 2^{n-2} elements.

3. For the function $g(x) = \frac{1}{2^n}\sum_{y \in t^\perp}(-1)^{x \cdot y}$, for any x, we have $g(x) = \frac{1}{2}(\delta_{x,0} + \delta_{x,t})$ where δ denotes Dirac delta.

   This result follows from (2) by partitioning $t^\perp$ into E_0 and E_1. Intuitively, this lemma shows how interference behaves over the entire set of orthogonal vectors $t^\perp$ and how the phases cancel themselves out.

4. Let y be the vector returned by Simon's subroutine, then the probability that it is orthogonal to some random vector $t \in \{0,1\}^n$ is bounded away from 1. Specifically, we have:

[5] A permutation is a bijection function with its domain and co-domain equivalent.

$$Pr_y[y \cdot t = 0] = \frac{1}{2}[1 + Pr_x[f(x) = f(x \oplus t)]]$$

This result can be derived from the final circuit state before measurement by applying (3) and Theorem 1 follows from it. Recall we wanted an even distribution for the expression $y \cdot x$ in eq 2 where y is the measured state and x is a random element in the pre-image of f. This result gives us the desired result when $\epsilon(f, s) >= Pr[f(x) = f(x \oplus t)]$ is bounded

5. Let $t \in \{0,1\}^n$ and $p_t = Pr_x[f(x) = f(x \oplus t)]$. Then the probability that all cn vectors measured from running Simon's subroutine will be orthogonal to t is equal to $(\frac{1+p_t}{2})^{cn}$.

   This result follows from (4) because we assume that the vectors u_i's obtained from Simon's subroutine are independent. Thus the probability cen be obtained by simply exponentiating (4). This completes the proof for Theorem 2.