# Quantum Improvements to Existing Cryptanalytic Techniques Against Symmetric Key Encryption Schemes

Yaelle Goldschlag

Daniel McVicker

CMSC 657

13 December, 2018

Daniel McVicker, Yaelle Goldschlag


**1. Intro** **2**

**2. Research Question** **2**

**3. Methodology** **2**

**4. Framing Cryptography** **3**

4.1 Information theoretically secure cryptography [Aaronson] 3

4.2 Computationally secure cryptography 4

**5. Threat Models** **5**

**6. Main Classes of Quantum Attacks** **6**

**7. Basis of Symmetric Key Cryptography** **7**

**8. Defining "Broken"** **7**

**9. Sample Attacks** **8**

9.1 Brute Force Search [Grover] 8

9.2 Grover Attack for Collision Finding [MVZJ] 9

9.3 Fixed Point Grover's [GLRS] 9

9.4 Linear Cryptanalysis [KLLN] 9

9.5 Differential Cryptanalysis [KLLN] 11

9.6 Slide Attack [BNS] 11

9.7 Meet-in-the-Middle Attacks [HS2] 12

**10 Conclusion** **13**

**References** **14**

Daniel McVicker, Yaelle Goldschlag


# 1. Intro

Most public-key encryption algorithms used today can be easily broken by quantum computers [Shor]. For symmetric-key algorithms, the extent to which quantum computers can reduce their security is not fully understood [BL]. Any $n$-bit key can be recovered in $O(2^{n/2})$ time using Grover's algorithm (as opposed to $O(2^n)$ time required for classical brute-force searches) [Grover]. Hence, a naive approach would suggest that doubling the key length would give a cryptosystem comparable security against quantum attacks as it previously had against classical attacks. There exist known quantum attacks against various symmetric-key cryptosystems with time complexity better than Grover's algorithm, where doubling the key size is insufficient to regaining the same standard of security [HS1].

Many quantum cryptanalysis against these symmetric-key cryptosystems have been ad-hoc. A more general methodology would make it easier to understand the capability of quantum attacks on symmetric key schemes. One common strategy is to look at a classical attack against a given cryptosystem, and then apply quantum improvements to applicable steps of that attack [KLLN].

# 2. Research Question

In this survey, we will review the existing literature on quantum attacks derived from classical cryptanalytic techniques. We will analyze the general techniques used in this category of attack and describe common approaches which could be applied towards new algorithms.

# 3. Methodology

We begin by exploring which attacks on symmetric key algorithms have been sped up with quantum, and how much the time complexity was improved. We categorize the attacks and encryption schemes by the quantum techniques used and the time complexity. We look specifically at whether the quantum attacks are within the quantum or query model, and whether the complexity is polynomial or exponential.

We then look at attacks on other symmetric key encryption schemes that have not been explored from the quantum perspective. We see if any of the quantum attacks can improve the brute force attacks on those schemes. We look specifically at whether the attacks can be improved to polynomial time complexity in the quantum query model or exponential time complexity in the classical query model.

We also look at if there are any symmetric key algorithms that have been conjectured to be quantum resistant, and see if we can extend that conjecture to other symmetric key algorithms.

# 4. Framing Cryptography

Symmetric key cryptography relies on two parties having a shared secret. This shared secret can be used to generate encryption and decryption keys. This scheme can be used as the building block for MACs and hash functions. Because this encryption is relied upon and built upon, we need clear guarantees of the security that this encryption provides.

Symmetric key cryptography is really the problem of a one way function, with a plaintext as input, and a ciphertext as the output. The goal is that the function $f(p) = c$, where $f$ denotes the application of key $k$, reveals no (or very little) information about $p$ from $c$. Another way of saying this is that it is very difficult to compute the inverse of $f$ -- that $f$ is computationally close to a one-way function.

## 4.1 Information theoretically secure cryptography [Aaronson]

An encryption scheme that is *theoretically secure* cannot be broken no matter how much time the attacker has. This is guaranteed by ensuring that any string is just as likely to be the plaintext as any other string.[1]

The first such scheme was first realized with the one-time pad in 1882. The one time pad takes a plaintext as a binary string and a random binary key of at least the length of the plaintext. The sender and receiver both have copies of this string.

1. Both the sender and the receiver have copies of key $k$.
2. The sender encrypts a message $p$ with key $k$ and sends it: $p \oplus k = c$
3. The receiver decrypts the message with key $k$: $c \oplus k = p \oplus k \oplus k = p$

The operation $p \oplus k$ will flip each bit of the original message with independent probability $\frac{1}{2}$. This will result in a new string, $c$, which is completely random. This scheme requires a *truly random* string of at least the length of the message. The encrypted message has only the randomness of the message, so if the key is distinguishable from random, the message will also be distinguishable from random. In addition, that random string can only be used once. If it is used twice, then information will be revealed about the key, differentiating it from random.

[1] Any valid scheme will reveal the maximum possible length of the plaintext.

Consider if the sender sends two messages to the receiver, encrypted with the same key $k$. This can reveal $p_1 \oplus p_2$ -- information which is not necessarily useful, but is no longer theoretically secure.

1. $p_1 \oplus k = c_1$
2. $p_2 \oplus k = c_2$
3. $c_1 \oplus c_2 = p_1 \oplus k \oplus p_2 \oplus k = p_1 \oplus p_2$

In 1945, Claude Shannon showed that *any* theoretically secure encryption scheme requires a truly random string of at least the length of the original message.

*Proof:* Every message must be equally likely to be the original message. This means that every message must be a possible plaintext for a given ciphertext. For a binary ciphertext of length n, there are $2^n$ possible input plaintexts. There cannot be more plaintexts than keys so, there must be $2^n$ possible keys, meaning that the length of the key must be at least the length of the plaintext. As mentioned above, the ciphertext is only as random as the plaintext, so this key must be truly random. And this key cannot be used more than once, because then the inputs together will be longer than $n$, necessitating the use of a longer key [Shannon].

### 4.2 Computationally secure cryptography

The key requirements for theoretically secure cryptography are infeasible, so cryptographers have settled for a lower standard. The goal of computationally secure cryptography is that the attacker cannot distinguish the ciphertext from a random string in polynomial time.[2] To do this, we use pseudo-random keys rather than random keys. A short pseudo random seed can be used to generate a longer random string of the same randomness (proof outside the scope of this paper), so a symmetric key encryption scheme based on this string can be just as safe as the problem of distinguishing the pseudo-random string from random.

It has been proven that the problem of one-way functions is of equivalent complexity to that of a random generator. The one way function itself should be vulnerable to a Grover-type attack which will give quadratic speed up, but not to a more effective attack. Modern symmetric key cryptography relies on -- and in the most fundamental case, is equivalent to -- these pseudo-random generators and one way functions. To bring this back to quantum, this is the reason why symmetric key cryptography seem safe: it is assumed (although not proven) that pseudo-random generators are hard to break. The symmetric key encryption schemes are based on this

[2] Related to the length of the string, with "distinguish" meaning distinguishing with non negligible bias.

problem, and the steps of the encryption that use this are resistant to the known categories of quantum attacks.[3]

# 5. Threat Models

Fundamental to any discussion on security is the *threat model*, a rigorous definition of what an adversary is capable of [KL]. For example, "known-plaintext" is a threat model where the adversary has access to some limited number of plaintexts along with their associated ciphertexts. Once the threat model is chosen, one can make claims about the security of the system against various types of attacks, e.g. "the best known chosen-ciphertext distinguishing attack against this cipher has a time complexity of $O(2^n)$". In accordance with Kerchoff's principle, the adversary is considered to have prior knowledge of every aspect of the cryptosystem (e.g. the encryption/decryption algorithms, and the probability distribution over possible plaintexts sent). Beyond that, there are many variations. Some commonly-discussed threat models include:

- *Ciphertext-Only*: The adversary has access to some limited number of ciphertexts, and nothing else
- *Known-Plaintext*: The adversary has access to some limited number of ciphertexts, as well as the corresponding plaintext for each ciphertext
- *Chosen-Plaintext*: The adversary has access to an *encryption oracle*, i.e. a black-box function which encrypts any plaintext queries provided by the adversary
- *Chosen-Ciphertext*: As in chosen-plaintext, but the adversary also has access to a decryption oracle

In the quantum setting, the only modification required for the ciphertext-only and known-plaintext threat models is the time complexity required for the adversary to locally solve certain computational problems (e.g. the quantum adversary is capable of factoring large numbers in poly-time, while a classical adversary is currently believed to be unable to do so).

There is no clear analogous threat models when an oracle is involved.[ Zhandry] is credited with formalizing two popular threat models for quantum oracles, which are commonly referred to as Q1 and Q2.

[3] Public key cryptography attempts to rely on one-way functions, but requires that these function have a backdoor. Because of this, most of these functions are fundamentally different from those used by symmetric key cryptography, and are not as "hard" to reverse as is it to identify pseudo-randomness.

- *Q1*: The adversary has access to local quantum computation, but oracles provided to the adversary accept only classical queries
- *Q2*: The adversary has access to arbitrary local quantum computations and oracles provided to the adversary accepts arbitrary quantum states as queries

Q2 is a strictly stronger threat model than Q1, so the security of a cryptosystem under Q1 is at least as strong as its security under Q2. It may seem sensible to demand security under the stronger model. However, while there exist examples of real-world scenarios in which an adversary's capabilities closely resemble each of the earlier discussed threat models, some have argued that the Q2 model may be unrealistically strong [HS1][KLLN]. In both classical and post-quantum cryptography, encryption schemes are concerned with with obfuscating classical, not quantum data. Furthermore, it is often assumed that the user is running classical computation rather than quantum, and the computation will be happening on a classical computer. It is likely that most real-world encryption oracles will be purely classical, preventing Q2 adversaries altogether. Because of this, both the Q1 and Q2 threat models are of interest to post-quantum cryptography.

# 6. Main Classes of Quantum Attacks

Current popular cryptographic algorithms rely on several "hard" classes of problems; common ones include: integer factorization, discrete logarithm, and elliptic-curve discrete log problem. All three of these problems become easier using Shor's algorithm. Symmetric key algorithms and hash functions are thought to be more secure. Grover's algorithm provides a quadratic speedup against both symmetric key encryption and hash functions, but this speedup can be mitigated by doubling the key size. More concerningly, there are several attacks which translate a scheme into Simon's problem, allowing for exponential speedup which no key length is long enough to make secure.

Many symmetric cryptanalysis techniques are based on a small set of problems that are assumed to be hard. Two of these problems are searching for collisions and multi-target preimages [CPS]. Quantum solutions to these two problems have significant performance increases against these assumptions.

# 7. Basis of Symmetric Key Cryptography

Classical symmetric cryptography assumes the following difficulty for a few problems.

- *Collision resistance:* Find two messages $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$. Analysis of the known classical algorithm relies on the birthday paradox, which says that this should be solvable in $O(2^{\frac{n}{2}})$.
- *Second preimage resistance:* Given $m_1$, find $m_2$ such that $H(m_1) = H(m_2)$. Exhaustive search solves this in $O(2^n)$.
- *Preimage resistance:* Given $h$, find $m$ such that $H(m) = h$. Exhaustive search solves this in $O(2^n)$.

Each of these problems have exponential speedup with quantum. These problems are the building blocks of symmetric key cryptography, and attacks also provide a speedup on more involved algorithms that are based on these problems.

# 8. Defining "Broken"

After deciding on a threat model, it's important to define what sorts of attacks count as "breaking" a cryptosystem. Typically, the security of a cryptosystem is presented in game-theoretic terms, showing that an adversary gains negligible advantage from a bounded number of queries [KL]. In these analyses, the strength of the adversary's advantage is measured in terms of a *security parameter*, which in practice is interpreted as the key length of the cryptosystem. More concretely, one can use these analyses to compute the expected time required for an adversary to perform a certain malicious action, e.g. "for a cryptosystem with key length *n*, a known-plaintext classical brute-force search can learn the key in $O(2^n)$ time. Some attacks we will talk about are:

- *Key Recovery*: The adversary learns the target's secret key
- *Message Recovery*: Given a ciphertext, the adversary learns the associated plaintext
- *Distinguishing*: Given either an encryption oracle or an oracle for a random oracle, the attacker determines with high probability which type of oracle it is
- *Forgery*: Under an authenticated cryptosystem, the adversary creates a valid ciphertext-tag pair

The standards for how strongly a cryptosystem must resist each type of attack is often dependent on the threat model. Notably, we will speak about the different standards often applied to Q1 and Q2 models in order for a system to be described as broken. Because of the relative strength of Q2, a "successful" attack under Q2 has a much higher standard than is required for an attack to be considered successful under

Q1. Because of this, there are cryptosystems for which there is a known successful attack under just Q1, just Q2, or both [HS1].

# 9. Sample Attacks

We present several classical attacks for which a stronger version has been shown under quantum threat models.

### 9.1 Brute Force Search [Grover]

The simplest "attack" against a cryptosystem is to exhaust the search space. Given a small number of known plaintext-ciphertext pairs, key recover can be rephrased in terms of solving preimages. With a key of length $2^n$, a plaintext *p,* and a ciphertext *c*, we fix *p* and interpret the encryption scheme as a function of the key, and try to solve for *k* in the equation $E_k(p)=c$. Although brute force attacks are only viable against cryptosystems with small keyspaces, they serve as a benchmark against which to compare more complex attacks.

Classically: By trying out out all $2^n$ possible keys, we are guaranteed to find a valid *k*, thus recovering the key. If there are multiple valid *k*'s for a given plaintext-ciphertext pair, in a well-designed cipher the number will be small. To find the correct key, we can test the candidate keys against another plaintext-ciphertext pair, continuing to iterate this process until only one valid key remains. Each iteration accepts negligibly many keys, and has negligible probability of failure, so this algorithm performs key recovery in $O(2^n)$ time.

Q1: Given *p* and *c* as in the classical case, we create an oracle for the function $f(k) = \{1 \text{ if } E_k(p) = c;\ 0 \text{ otherwise}\}$, satisfying the formulation for Grover's problem. Thus, we can then apply Grover's algorithm to *f*, and the result will likely be a key. Given the search space of $2^n$ possible keys, Grover's algorithm recovers the key in $O(2^n)$ time. As in the classical case, further plaintext-ciphertext pairs can guarantee success while adding negligible extra time to the algorithm.

### 9.2 Grover Attack for Collision Finding [MVZJ]

Without any modifications, Grover's algorithm can be used to find a collision in a hash function in $O(\sqrt{n})$, where n is the length of the original string. Grover's algorithm can be combined with the birthday paradox to improve this to $O(\sqrt[3]{n})$. This is done by creating a table of size cube root n, and searching for a collision in that table, which will take $O(\sqrt[3]{n})$. If there is no collision, then each value in the table has a collision with a value

not in the table. We are searching for cube root n elements over $1 - \sqrt[3]{n}$ elements, which is solvable using Grover's in $O(\sqrt[3]{n})$ [MVZJ].

### 9.3 Fixed Point Grover's [GLRS]

Grover's algorithm provides quadratic speedup, doing an unstructured database search of n elements in O(sqrt n). Grover's algorithm find one element from n elements in O(sqrt n), a search that would classically take O(n). A variation of this problem involves searching for one of m elements from n elements. Classically, this takes O(n/m). With quantum, this problem can be solved when m is known in O(sqrt (n/m)) by doing sqrt (n/m) iterations.

If m is unknown, there are a number of possible solutions. One can first count the number of solutions, and then do the usual algorithm. One could also apply an adaptive schedule where the Grover operator, instead of operating by the same amount with each iteration, is replaced by an operator that rotates a different amount based on the iteration index. This causes the quantum state to converge to a bounded region, which can then be measured. This bounded region is called the fixed point, leading to the algorithm name of Fixed Point Grover's [GLRS].

### 9.4 Linear Cryptanalysis [KLLN]

Consider an equation of the form $m_1+...+m_n+...+c_{i+...}+k_i+...+k_n = 0$, where $m$ is a plaintext input, $k$ is the key, and $c$ is the corresponding ciphertext output. If such an equation were known to be an invariant of the cryptosystem, then knowledge about plaintext-ciphertext pairs can be used to gain information about the bits of the key. For an ideal cryptosystem, any such linear equation would hold true for exactly half of all possible inputs, thus granting no information. A more practical goal is to find several equations which hold true for significantly more than half (or significantly less than half) of all possible inputs. For an equation which is holds with probability $(1+e)/2$, it can be shown that the sign of $e$ can be determined with high probability from $O(1/e^2)$ plaintext-ciphertext pairs

Classical distinguisher: It can be shown that, with high probability, a linear equation which holds w/ probability $(1+e)/2$ is expected to hold for $>e/2$ of randomly sampled inputs to the encryption oracle, and $<e/2$ of randomly sampled inputs to a random oracle. This likelihood reaches statistically significant levels after $O(1/e^2)$ samples, so a known-plaintext distinguisher can be built which runs in time $O(1/e^2)$

Q1 distinguisher: Due to the nature of the distinguishing problem, Q1 is rarely known to have any advantage against classical distinguisher attacks. Statistical tests for

differentiating distributions tend not to be computationally intensive, and the bottleneck for most distinguishing attacks is the number of samples. Hence, Q1's sampling situation is no different than in the classical case, and the adversary does not benefit much from local quantum computation, so Q1 is not known to provide speedup in distinguisher attacks.

Q2 distinguisher: Like the classical distinguisher, except we determine the distribution using the "quantum counting" variant of the amplitude amplification algorithm. In this case, the distinguisher is able to distinguish the encryption oracle from the random oracle using only time $O(1/e)$

Classical key recovery: With $L$ independent linear equations, a chosen-plaintext adversary can, from $O(L/e^2)$ plaintext-ciphertext pairs, determine $L$ bits of information about the key by performing $L$ iterations of the distinguisher routine and observing the sign of the bias, cutting down the search space by a factor of $2^L$. Thus, an $n$-bit key can be recovered in $O(L/e^2 + 2^{n-L})$ time.

Q1 key recovery: Similar to the classical case, the adversary makes $O(L/e^2)$ queries to learn bits of the key. However, the rest of the key can be learned locally with Grover's algorithm, cutting the time complexity down to $O(L/e^2 + 2^{(n-L)/2})$.

Q2 key recovery: Similar to Q1, except the adversary uses the Q2 distinguisher subroutine to learn the bias of the linear equations, taking $O(L/e)$ time, thus performing full key recover in time $O(L/e + 2^{(n-L)/2})$.

### 9.5 Differential Cryptanalysis [KLLN]

Suppose we have an invariant of the form $E_k(x+p)=E_k(x)+q$. For any fixed $p,q$, in an ideal cipher this invariant would hold for half of all possible $x$'s. The goal is to find $p,q$ such that the equation holds for either much greater than or much less than half of all possible $x$'s.

Classical Distinguisher: Given $p,q$ s.t. our differential invariant holds/does not hold w/ probability at least $2^{-k}$, then we expect to find an $x$ satisfying the invariant once every $2^k$ samples, or twice in $2^{k+1}$ samples. However, for a random oracle, we expect the invariant to hold w/ probability $2^{-n}$. Thus, is $k$ is significantly less than $n$, there is extremely low probability of more than one $x$ in $2^{k+1}$ samples satisfying the invariant under the random oracle. Hence, a known-plaintext classical adversary can execute a distinguishing attack against the cryptosystem with time complexity $O(2^{k+1})$.

Q2 Distinguisher: Given $p, q$ as above, we create a wrapper oracle which takes an input $x$, the outputs 1 if the invariant holds (which, we note, requires 2 queries to the

encryption oracle), or 0 otherwise. Then, we can apply Grover's algorithm to the wrapper oracle in order to find an input following the invariant with just $O(2^{k/2+1})$ queries to the encryption oracle. However, if the oracle is a random oracle, then after only $O(2^{k/2+1})$ queries the resulting state will still have most of its probability in the "0" subspace, making the output likely to *not* be an input which follows the invariant. Thus, a known-plaintext Q2 adversary can execute a distinguishing attack against the cryptosystem with time complexity $O(2^{k/2+1})$.

Key Recovery: Differential key recovery attacks are similar to linear key recovery attacks. Using $L$ differential invariants, we perform $L$ rounds of the appropriate distinguisher routine, followed by a local brute force search over the remaining keyspace.

### 9.6 Slide Attack [BNS]

While increasing the number of rounds can increase the security of a cipher, the slide attack is a demonstration that certain weaknesses do not disappear with any number of additional rounds. The typical example of one such weakness is when a single round is vulnerable to a small known-plaintext attack, for example in a sequence of keyed permutation. Throughout, a single round will be denoted as $R(x) = P(x + k)$, where $P$ is the permutation and $k$ is the round key.

Classical: With $O(2^{n/2})$ plaintext-ciphertext pairs $(p,c)$, we expect at least one pair $p1$ and $p2$ such that $p2=R(p1)$. Consequently, $c2=R(c1)$. Thus, for each pair which is one round apart in the full cipher, we actually get two pairs of plaintext-ciphertext for the round function. This process is repeated until enough pairs are known in order to recover the key from the round function.[4]

Q2: Note that the slid pair we need is described by the invariant $E_K(R(x))=R(E_K(x))$. We can rewrite this as:

$$f(b\|x) = \begin{cases} R(E_K(x)) + x & b = 0 \\ E_K(R(x)) + x & b = 1 \end{cases}$$

Here, note that $f(b||x) = f((b||x) + (1||k))$, thus satisfying the criteria of Simon's problem. Thus, by using Simon's algorithm on this oracle, we can perform the slide attack with only $O(n)$ queries.

[4] For simplicity, we assume that the slid pair are one round apart, and all round keys are the same. Similar variations exist for pairs any length apart and varying round keys as long as the round keys are predictable.

### 9.7 Meet-in-the-Middle Attacks [HS2]

Rather than adding more rounds to an encryption scheme, another strategy for increasing security is to repeat the entire algorithm with different keys. This was the strategy that led to the creation and adoption of the TripleDES encryption algorithm (which, as the name would imply, is the DES algorithm run three times). However, the meet-in-the-middle attack shows that this method does not achieve the same amount of security as would be expected from the total length of the keys involved.

Classical: Suppose the cryptosystem is a weak scheme run twice. To break it, the adversary performs a brute-force attack on the first key alone, recording the output. Then, the adversary runs a brute-force attack *decrypting* from potential second keys, again recording the results. Between the two lists, there will be a collision at the correct values for both keys. Thus, rather than attacking every pair of keys, the adversary is able to attack the round keys individually, hence double-encryption has no more strength than single-encryption.

Q1: The matching algorithm can be done using local quantum searches, bringing the complexity down to $O(2^{n/2})$

Q2: Rather than computing each possible key on both sides, we instead use a quantum counting distinguisher (similar to the ones used in the linear distinguishing problem). From there, we proceed as in Q1. The advantage in this case is not in time complexity (the complexity roughly matches the Q1 case), but instead uses vastly fewer space resources (completing the necessary data for the matching table in $O(1)$ memory)

# 10 Conclusion

Unlike most public-key schemes, much of existing symmetric-key cryptography schemes are not trivially broken by widely-known quantum algorithms. The extent to which the quantum threat model breaks current symmetric-key cryptosystems is not yet fully known, and is widely dependent on what quantum threat model is shown to be realistic. By making improvements to existing cryptanalysis techniques, we create a strong starting point from which to develop more powerful quantum attacks against these schemes.

# References

[Aaronson]: Scott Aaronson. Quantum Computing since Democritus. Cambridge: Cambridge University Press. pp. 97-105.

[BL]: Daniel Bernstein, Tanja Lange. *Post-Quantum Cryptography: Dealing with the Fallout of Physics Success*. IACR Cryptology ePrint Archive 2018: 314 (2018)

[BNS]: Xavier Bonnetain and María Naya-Plasencia and André Schrottenloher. *On Quantum Slide Attacks*. IACR Cryptology ePrint Archive 2018: 1067 (2018)

[CPS]: Andre Chailloux, Maria Naya-Plasencia, Andre Schrottenloher. An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography. https://eprint.iacr.org/2017/847.pdf

[DDW]: Xiaoyang Dong, Bingyou Dong, Xiaoyun Wang. *Quantum Attacks on Some Feistel Block Ciphers*. IACR Cryptology ePrint Archive 2018: 504 (2018)

[GLRS]: Markus Grassl , Brandon Langenberg , Martin Roetteler, Rainer Steinwandt. Applying Grover's algorithm to AES: quantum resource estimates. https://arxiv.org/pdf/1512.04965.pdf

[Grover]: Lov K. Grover. *A fast quantum mechanical algorithm for database search*. In Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), pages 212–219. ACM, 1996.

[HS1]: Akinori Hosoyamada, Yu Sasaki. *Cryptanalysis against Symmetric-Key Schemes with Online Classical Queries and Offline Quantum Computations*. IACR Cryptology ePrint Archive 2018: 977 (2017)

[HS2]: Akinori Hosoyamada, Yu Sasaki. *Quantum Demiric-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions*. IACR Cryptology ePrint Archive 2018: 1229 (2018)

[KL]: Jonathan Katz, Yehuda Lindell (2007). *Introduction to Modern Cryptography: Principles and Protocols*. Boca Raton: Chapman and Hall/CRC. OCLC 893721520.

[KLLN]: Marc Kaplan, Gaetan Leurent, Anthony Leverrier, and Marıa Naya-Plasencia. Quantum differential and linear cryptanalysis. arXiv:1510.05836.

[MVZJ]: Vasileios Mavroeidis, Kamer Vishi, Mateusz D. Zych, Audun Jøsang. The Impact of Quantum Computing on Present Cryptography. https://arxiv.org/pdf/1804.00200.pdf

[Shannon]: Claude Shannon (1949). Communication Theory of Secrecy Systems (PDF). Bell System Technical Journal. USA: ATT Corporation

[Shor]: P. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput., 26 (5), 1997, pp. 1484–1509. http://dx.doi.org/10.1137/s0036144598347011.

[Zhandry]: Mark Zhandry. How to construct quantum random functions. In 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, pages 679–687. IESEE Computer Society, 2012.