Quantum Computing: Effects on Cryptography

Quantum computing was first proposed as a concept of research by Richard Feynman in 1982 and its primary aim is to utilise the extraordinary properties exhibited by matter at a sub-atomic (or quantum) level. Many large corporations such as IBM, Google and Amazon are the current leaders of the field, contributing millions of dollars each to research and development. There are many promising applications for any outcomes of these experiments as a quantum chip is predicted to run significantly faster than a classical CPU, meaning it can efficiently solve multiple currently complex problems. However, such a considerable increase in processing power also has negative effects, for example, it can perform much stronger attacks to our current systems, leaving them vulnerable to unwanted hackers. Therefore, these developments have a critical impact on the world of cryptography. The algorithms used to encrypt our current data and keep our systems safe are based on complicated mathematical problems - for example, a common encryption method is called RSA (Rivest-Shamir-Adleman) and it is used in secure data transmission by many protocols (eg. Secure Shell (SSH) and SSL/TLS). The algorithm works on the idea that it is easy to compute the product of two large prime numbers, but difficult to factorise the outcome back to its prime factors and would take a modern computer many years to calculate. The public and private keys used to encrypt data are then built using this large number. However, a quantum computer can factorise this number much faster, using Shor’s algorithm, which leaves all data encrypted using RSA insecure. Many other cryptographic algorithms fall victim to the same problem. Two possible solutions which are presently being explored are Quantum Key Distribution and Post-Quantum Cryptography; the former explores using quantum phenomenon to perform the key exchange without any unwanted interception and the latter uses mathematical algorithms, which cannot be unravelled by a greater amount of processing power.

Quantum Key Distribution (QKD) is a secure method of communication which enables two parties to produce a shared secret key, which can be used to encrypt and decrypt messages. The cryptographic protocols use the properties of quantum mechanics, such as quantum states, superposition and entanglement, ensuring that if any third party attempts to eavesdrop and intercept the communications, the involved parties would be able to detect it. On a quantum level, particles are inherently uncertain - they can exist in more than one state or place at one time - so their properties are impossible to predict. They can be measured randomly based on their polarities (or spins), which serve as binary counterparts. However, when measured their state is altered significantly, preventing the process from going undetected and any further attempts would not produce the same results. Additionally, particles cannot be completely cloned, only partially, inhibiting an unwanted third party from creating another version of the information to read themselves. These properties make the quantum level ideal to use for secure communications. The most common protocol for QKD is BB84 and many others which exist are similar variations or methods based on the same concept. BB84 can be split into two separate phases: the quantum transmission to form a secret key and the classical post-processing, which asserts that the key shared between the parties is the same. This process can be depicted using the conventional parties: Alice and Bob. First, in the quantum stage, Alice generates a random sequence of bits and a random sequence of bases. She uses the bases to encode her bits into qubits and transmits these over a quantum channel to Bob. He also generates a random sequence of bases and uses these to measure the information he receives, keeping his results private. Then, in the post-processing stage, they each share the bases they used over the classical channel and discard any bits where their choice of bases differed. With the remaining bits, Alice and Bob sample a few to compare and if they align, they can use the rest as their shared secret key. However, if an unwanted third-party Eve intercepts any of the shared information, these samples would not match and Alice and Bob would be forced to discard their keys and restart, although this could also be due to noise in the channel or many other factors, which Alice and Bob cannot distinguish between.

Although BB84 was the first proposed protocol, there are many other versions created after to resolve problems which arose. An example is B92, which only uses two states of polarisation - as opposed to 4 like BB84. In the first phase, Alice chooses a sequence of bits to encode into corresponding qubits and Bob chooses a sequence of bits to encode into corresponding bases. Alice then sends her chosen bits and Bob measures them in his chosen bases. Another example is SARG04, a version of BB84 developed to be more robust against photon-number splitting attacks. Here, in the classical phase, Alice doesn't announce her chosen bases, but instead two possible states for each qubit, keeping to herself which is correct. Bob then determines whether zero, one or both possibilities align with his measurement. If he cannot be sure which is correct, they discard the bit, otherwise the bit is used in the sifted key. If Eve tries to intercept, she cannot obtain reliable information without a store of many photons, which would make her presence easier to detect. While these are similar to BB84, the final protocol, E92, relies on the properties of quantum entanglement, distinguishing it from the rest. Alice and Bob are given entangled particles and the two parties randomly choose a basis to measure them in. This is repeated over a sequence of bits and then they each share the bases they use over the classical channel. If either party didn't register a bit previously, the corresponding information is disregarded at this point. Otherwise, if they used different bases, they can use the information to determine if an eavesdropper was intercepting on their quantum channel. Any bits remaining can be used as their shared secret key. All these protocols are only a partial solution though, as QKD doesn't provide a means to authenticate the transmission source - this requires asymmetric keys or pre-placed keys. There are many other issues with QKD as well. It is much more expensive; it requires special purpose equipment which cannot be easily integrated as it is hardware based; it increases the risk of insider threat and is susceptible to denial of service attacks.

Post Quantum Cryptography (PQC), on the other hand, involves a list of quantum-safe mathematical algorithms, decided by the US National Institute of Standards and Technology (NIST), which are set to be the new cryptographic standards. These include CRYSTALS-Kyber for public-key encryption and CRYSTALS-Dilithium for digital signatures, both of which are lattice-based forms of cryptography. The lattices are formed by different combinations of vectors sets and along with the principles derived from two problems, these can be used to form a cryptographic system. The first problem is called the shortest vector problem, which tackles the issue that it is difficult to determine the shortest vector in a lattice. This is equivalent to the problem of finding the closest vector to a given one and finding the basis for a lattice with the shortest possible vectors. The second problem is called Learning With Errors (LWE) and involves removing (or detecting) noise within a data set. The cryptographic system is built as follows: Alice generates a secret vector s_1 and a small error vector e_1 and similarly, Bob generates his own vectors s_2 and e_2. They are both then given a square array A and Alice calculates $b_1 = s_1 \cdot A + e_1$ while Bob calculates $b_2 = s_2 \cdot A + e_2$. Alice throws away e_1 and sends Bob her b_1, from which he calculates the key: $k_2 = b_1 \cdot s_2 = s_1 \cdot A \cdot s_2 + e_1 \cdot s_2$. Finally Bob sends Alice his b_2 and she uses it to calculate her key: $k_1 = b_2 \cdot s_1 = s_2 \cdot A \cdot s_1 + e_2 \cdot s_1$. The secret vectors and error values were given small magnitudes and the keys only differ by $e_1 \cdot s_2$ and $e_2 \cdot s_1$, therefore using truncation (or other forms of error correction), the keys can be considered equal. This exchange method is very similar to the Diffie-Hellman protocol, except for the slight probability of getting mismatched keys, which increases with the number of calculations and the non-commutative operations used, meaning the order of calculations is significant. Safe versions of this protocol require using large matrices and generating large keys. To reduce this, the matrix operations are replaced with polynomial functions, which can be reduced using modular arithmetic, forming rings - so this is called Ring Learning with Errors (RLWE).

Kyber creates a key encryption algorithm from the LWE-based Diffie-Hellman exchange described above. Alice generates her public key b_1 and Bob calculates and uses his key k_2 to encrypt his message (cyphertext $c = k_2 + m$). Alice can decrypt this message using Bob's b_2 and her s_1: $m = c - k_1$ and she can shift it to remove any errors. Dilithium uses different parameters to Kyber, but still generates public and private keys. However they are used differently for digital signatures - the signer generates a random vector y and calculates $w = \text{HighBits}(A \cdot y)$ and $c = \text{HASH}(m \,||\, w)$ and $z = y + c \cdot s$. Here, s is the private key, m is the message to sign and the coefficients of z are verified to be below a given threshold (otherwise the value of y is reassigned). Values z and c are sent as the signature and the recipient verifies these by calculating $w' = \text{HighBits}(A \cdot z - c \cdot b)$, where b is the public key. If $c = \text{HASH}(m \,||\, w')$, then the signature is valid. This demonstrates how lattices are used to build cryptographic systems and are used in these protocols, however, they also introduce new issues. The problems that arise include vulnerability to error injection attacks, the error processing being prone to leaking side channel data and the reliance on secure random matix generation. Although these examples are lattice-based, the other algorithms selected by NIST rely on other mathematical properties, such as hashes, codes and isogenies.

Using these protocols, along with many others. quantum cryptography promises to solve problems predicted to arise in the future due to the discovery and research into quantum computing. Although modern quantum computers are still small and error-prone, the impending reduction in security is still relevant now as it threatens to expose all information being encrypted using current standards. NIST suggests that over the next two decades, quantum computing has the potential to scale significantly, enough to compromise currently used key encryption methods, so these new security measures are important to protecting all sensitive information in the very near future. Although cryptography is one area of research greatly impacted by the age of quantum computing, it is only a drop in the ocean. There are many other uses and threats posed by quantum computers, which simultaneously need to be explored and also protected against. The new capabilities of technology will also benefit chemical and pharmaceutical research, with the potential to predict molecular properties and enhance the quality of medicine development processes. Additionally, they can help financial institutions optimise investment portfolios and accurately detect fraud or anomalous transactions. Other industries such as aerospace or robotics can optimise their systems and use the new technology to improve their current designs. Overall, quantum computing will hopefully have a positive impact on many sectors of research and so will be contributing to society's developments in a few years. However, this makes the threats it poses much more important to prevent, using algorithms and protocols proposed by QKD and PQC to achieve this.