2016 3rd Asia-Pacific World Congress on Computer Science and Engineering

The Need for Quantum-Resistant Cryptography in Classical Computers

Mohammed Farik *and* Shawkat Ali

{mohammedf, shawkata}@unifiji.ac.fj

***Abstract* – In this review paper, we present reasons the current best cryptographic algorithms will fail classical computer security in post-quantum era. The presented security gaps outline the need to develop quantum-resistant cryptographic functions and algorithm for classical computers, with a few novel recommendations to the effect. Therefore, we believe this paper will enlighten and generate interest in post-quantum cryptography research.**

Keywords *– classical computing, hash, public-key cryptography, quantum computing, quantum-resistant*

I. INTRODUCTION

We live in a connected world and rely heavily on secure internet services for email, social networking, web search, cloud computing, e-commerce, and bill payment, amongst hundreds others [1]. For example, the https protocol uses 128-bit encryption at Secure Socket Layer (SSL) to protect web traffic for banking, e-commerce, email, amongst other services. A look at http://map.norsecorp.com/#/ website (Fig.1) shows the extent of cyber-attacks on a global scale [2]. As can be seen in this live attack map, the 65535 ports that support various computing services such as email, https amongst hundreds others are under constant cyber-attack.

Figure 1. Norse Live-Attack Map

Much of the increased attacks can be attributed to the fact that computers have become increasingly powerful in terms of speed and capability. Table 1 compares some types of computers and their processors that are currently making headlines all over the world. While classical computers perform as many computations at the same time as there are cores in its processor, quantum computers perform as many computations exponentially as there are quantum bits (qubits) in its processor. Quantum effects such as superposition of bits 0 or 1; parallelism [3], entanglement [4], [5], and quantum annealing give D-Wave X2 quantum computer this enormous capability. For example, D-Wave X2 quantum computer has a single processor, but has 1000 qubits that can perform 2^{1000} calculations simultaneously [6]. In comparison, the fastest supercomputer "Sunway TaihuLight", has 10,649,600 processor cores capable of performing only as many computations at the same time [7].

TABLE 1. COMPUTER/PROCESSOR SPEEDS

Systems	Processor Core	Frequency
Samsung Galaxy S7 smartphone	8 Core Snapdragon 820, Exynos 8890, 64-bit chipset	2.3+ GHz [8]
Desktop PC	Intel Core i7-7Y75 (7th Gen. Processor)	3.6 GHz [9]
Sunway TaihuLight Supercomputer	10,649,600 cores, 1.45GHz	93,014 TFlop/s [7]
D-Wave X2 Quantum Computer	1 x 1000 qubit CPU	2^{1000} simultaneous computations [6]

D-Wave's performance advantage suggest future quantum computers will be even more powerful and solve many of the physical world's currently difficult quantum mechanical challenges in the areas of artificial intelligence, machine learning, image recognition, materials modeling, drug discovery, and search and optimization faster and better than today's fastest supercomputers.

However, the same capability of quantum computers will open up the Pandora's Box in the face of classical computer cryptography. Cryptography is by far the best technique implemented to protect information for confidentiality and integrity in classical computers. Modern cryptography makes use of mathematical theory and computer science practice when designing computational algorithms. Any chosen algorithm should be computationally secure, meaning computationally difficult to break in practice by any attacker. Many cryptographic protocols are based on the difficulty of factoring large composite integers, prime numbers, or a related problem. With the presence of quantum computers such as D-Wave and personal quantum computers (PQCs) in the attack vector in the near future, there is threat that our current cryptographic defenses will not be able to provide adequate security.

Accordingly, this review paper intends to discuss the gaps that exist in hash, encryption, digital signature, and key exchange algorithms for post-quantum use in classical

---

- Mohammed Farik is a PhD candidate and a Lecturer in Information Technology at The University of Fiji. E-mail: mohammedf@unifiji.ac.fj
- Shawkat Ali is a Professor in Information Technology at The University of Fiji. E-mail: neerajs@unifiji.ac.fj

978-1-5090-5753-5/16 $31.00 © 2016 IEEE
DOI 10.1109/APWC.on.CSE.2016.25

computers and make recommendations for improvements as per NIST's Post-Quantum Crypto Project [10].

In the following Sections, Section II explains foundation of current cryptographic algorithms. Section III addresses the security gaps that open up in classical computers due to emergence quantum computers. Section IV sketches some recommendations as solutions and finally Conclusions in Section V.

II. CLASSICAL COMPUTING CRYPTOGRAPHIC ALGORITHMS

Currently, the best way to ensure security in all digital infrastructure such as network hardware, communication protocols, and software is by implementing cryptographic functionalities such as encryption, hash functions, digital signature, and key exchange [10],[11]. This Section discusses three classes of cryptographic algorithms – namely, hash functions, symmetric-key algorithms and asymmetric-key algorithms, and the mathematical basis for their acceptance.

A. Mathematical Basis

The strength of all cryptographic algorithms is based on difficult mathematical problems that generate codes which unauthorized people will not be able to easily break. Today, in classical computers, the mathematical theory of Integer factorization is used to strengthen public-key cryptography systems because it is computationally difficult to factorize large integer in classical computers, particularly if the integer is a product of two 300-digit (2400-bit) prime numbers.

B. Cryptographic Hash Function

Cryptographic hash function uses a mathematical algorithm that converts a message (input) of any length to a hash value (digest) string of fixed bit-size in a one-way operation (Fig.2) that is impossible to reverse [12], [13]. It is used in information security applications such as digital signatures, message authentication codes (MACs), data indexing in hash table, in fingerprinting, and as checksums [13].

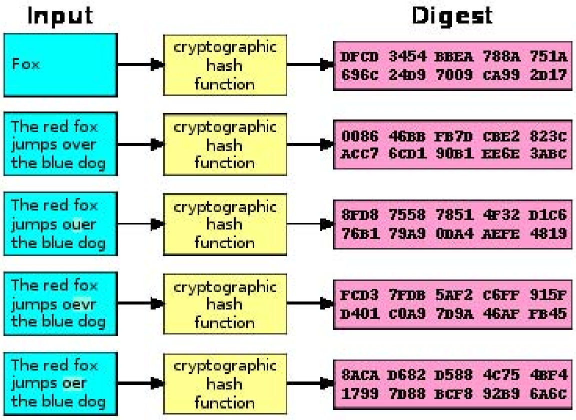

Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Figure 2. Use of Cryptographic hash function [13]

A perfect cryptographic hash function has four characteristics. One, it should be quick in calculating the digest from the input. Two, the digest cannot be used to get back the original input. The only way to get the input is by attempting a brute-force search of possible inputs to derive a match. Three, any change in input changes the message so severely that the new digest will be in no way correlated to the old. Fourth, it is impossible to find two different inputs derive the same digest [13].

A cryptographic hash function should resist all known cryptanalytic attacks such as pre-image attack, second-pre-image attack, and collision attack.

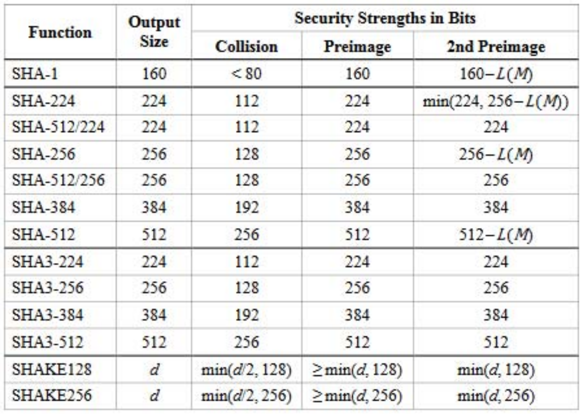
Pre-image resistant hash function is one where for a computed digest h it is difficult to find any input (m) such that $h=hash(m)$. If it is not difficult, then hash function is vulnerable to pre-image attack [13].

Second pre-image resistant hash function is one that when given an input m_1, it is difficult to find different input m_2 such that $hash(m_1)=hash(m_2)$. If it is not difficult, then hash function is vulnerable to second pre-image attack [13].

Collision resistant hash function is one where it is difficult to find two different inputs m_1 and m_2 such that $hash(m_1)=hash(m_2)$. For collision resistance, hash value should be twice as long as those required in second-pre-image resistance attacks. If it is not long enough, birthday attacks will find collisions [13], [14].

Secure hash algorithms (SHAs) are specified in FIPS180-4 [14] and FIPS202 [15] as recommended hash functions. Table 2 shows the security strength of SHA-1, SHA-2, and SHA-3 functions in classical computing [14], [15].

TABLE 2. SECURITY STRENGTHS OF SHA-1, SHA-2, AND SHA-3 FUNCTIONS [15]



Function	Output Size	Security Strengths in Bits		
		Collision	Preimage	2nd Preimage
SHA-1	160	< 80	160	$160-L(M)$
SHA-224	224	112	224	$\min(224, 256-L(M))$
SHA-512/224	224	112	224	224
SHA-256	256	128	256	$256-L(M)$
SHA-512/256	256	128	256	256
SHA-384	384	192	384	384
SHA-512	512	256	512	$512-L(M)$
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512
SHAKE128	d	$\min(d/2, 128)$	$\geq \min(d, 128)$	$\min(d, 128)$
SHAKE256	d	$\min(d/2, 256)$	$\geq \min(d, 256)$	$\min(d, 256)$

For a message that is less than 2^{64}-bits, SHA-1, SHA-224 and SHA- 256 hash algorithm is applied. For a message less than 2^{128}-bits, SHA-384, SHA512, SHA-512/224 and SHA-512/256 hash algorithm is applied. SHA-3, the most recent hash algorithm was released by NIST in 2015 as FIPS202 [14], [15]. SHA-3 is a family of four cryptographic hash functions (SHA3-224, SHA3-256, SHA3-384, and SHA3-512) and two extendable-output functions (XOFs), namely SHAKE128 and SHAKE 256 [15].

These SHAs are also built-in as part of many other cryptographic algorithms such as digital signature algorithms as detailed in FIPS186-4 [16], keyed-hash message authentication codes (HMAC) as detailed in FIPS198-1 [17], and in the generation of random number bits [14].

Some popular cryptographic hash functions such as HMAC are susceptible to length-extension attacks. If given *hash(m)* and *len(m)* but not *m*, an attacker can chose an appropriate *m'* to concatenate and calculate *hash(m||m')* [13].

C. *Symmetric-Key Algorithms versus Asymmetric Key Algorithms*

Symmetric-Key algorithms are also known as secret-key algorithms as they use the same key for both encryption and decryption purposes (Fig.3).

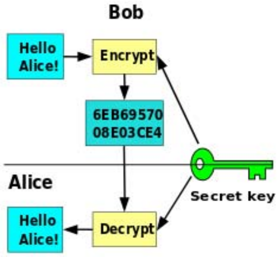


Figure 3. Symmetric key encryption [18]

Here, Alice can encrypt and send a message to Bob to decrypt Bob can encrypts and send a message for Alice to decrypt, using the same shared key. Symmetric key algorithms provide four functionalities. One, they can provide data confidentiality (privacy). Two, it can generate and validate a message authentication code (MAC). Three, it is used key-establishment process. Four, it can generate deterministic random numbers [12]. Currently, the strongest and recommended Symmetric key algorithm for encryption and decryption purposes is Advanced Encryption Standard (AES) [12].

Asymmetric-key algorithms, also known as public-key algorithms, use a pair of keys – private-key and public-key, for encryption and decryption purposes (Fig. 4). Procedures include [19]:

(a) Deciphering enciphered message M yields M, as

$$D(E(M) = M.$$

(b) It is easy to compute both *D* and *E*.

(c) Even by publically revealing *E*, public cannot find any easy way to compute *D* efficiently.

(d) If the message *M* is first deciphered and then enciphered, *M* is the result, as

$$E(D(M) = M.$$

In Fig. 4, for Alice to receive an encrypted message from Bob that she can understand, Alice has to give her public-key – E_A *to* Bob. Bob will encrypt the plain text message using Alice's public key – $E_A(M)$ and send to Alice. Alice will have to decrypt the message using her private-key – $D_A(E_A(M))$ to get the plain text message – M. Asymmetric-key algorithms can be used to compute digital signatures, and to establish cryptographic keys [12].

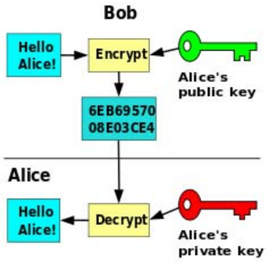
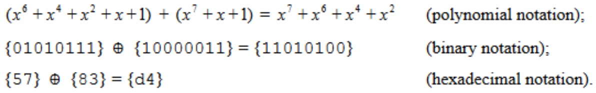


Figure 4. Asymmetric key encryption [18]

D. *Advanced Encryption Standard (AES)*

FIPS-197 discusses AES algorithm in detail [20]. In brief, AES uses sequences of 128 bits for input and output. Block length = 128 bits, $0 \leq n \leq 16$. Its cipher key contains 128, 192 or the strongest 256-bit sequence [20]. AES-256 makes 14 repetitions of transformation rounds in the matrix that convert plaintext (input) into cipher text (output), and vise-versa which are detailed by FIPS-197 [20] and ISO/IEC 18033-3 [21].

Further, AES performs polynomial calculations on input bytes that are represented as finite field elements as [20]:



$(x^6+x^4+x^2+x+1) + (x^7+x+1) = x^7+x^6+x^4+x^2$ (polynomial notation);

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (binary notation);

$\{57\} \oplus \{83\} = \{d4\}$ (hexadecimal notation).

E. *Digital Signatures*

A digital signature detects unauthorized modifications to data (integrity), authenticates identity of the signatory, and proves to a third-party that signature was generated the claimed signatory (non-repudiation) [16], [19]. Rivest *et.al* explains use of digital signature in the following example [19].

For Bob to send Alice a signed message *M* in a public-key cryptosystem, he first has to compute his signature *S* for the message *M* using D_B, such that:

$$S = D_B(M).$$

Bob then encrypts *S* using E_A (for privacy), and sends the result $E_A(S)$ to Alice. He does not need to send *M* because can be computed from *S*.

Alice has to first decrypt the cipher-text with D_A to obtain *S*. She presumes the sender is Bob, so she extracts the message with the encryption procedure of the sender, in this case E_B such that:

$$M = E_B(S)$$

Alice now possesses a message-signature pair (M, S) that has properties similar to those of a signed paper document. Hence, Bob cannot deny having sent Alice this message, because Alice could not have created $S= D_B(M)$.

So, Alice can convince a judge that $E_B(S) = M$, as she has proof that Bob signed the document.

Also, Alice cannot modify M to a different version M', as to do that she will also have to create the corresponding signature $S' = D_B(M')$.

Therefore, Alice has received a message signed by Bob, which she can prove that Bob has sent, but which she cannot modify [19].

FP186-4 details algorithms and methods for generating, verifying, and validating digital signature. FIPS186-4 approved the use of three algorithms for digital signature generation, verification, and validation purposes – Digital Signature Algorithm (DSA), Rivest-Shamir-Adleman (RSA), and The Elliptic Curve Digital Signature Algorithm (ECDSA) [16]. Fig. 5 shows that digital signature algorithms also rely on built in hash algorithms to determine data lengths for digital signature computation [12].

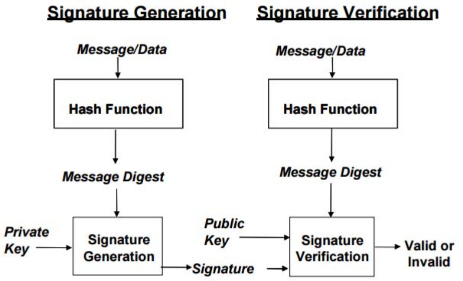


Figure 5. Hash use in Digital Signature Process [16]

F. Digital Signature Algorithm (DSA)

DSA technical specifications such as criteria for the generation of domain parameters, for the generation of public and private key pairs, and for the generation and verification of digital signatures are detailed in FIPS186-4 [16]. DSA key sizes mentioned are 1024, 2048, and 3072 bits while the output digital signatures are of 320, 448, or 512 bits [12].

G. Rivest-Shamir-Adleman (RSA)

Authors Rivest, Shamir, and Adleman detail the RSA algorithm in their 1978 paper, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" [19].

RSA encryption (E) and decryption (D) algorithms are mathematically as [19]:

$$C \equiv E(M) \equiv M^e \text{ (mod } n\text{), for a message } M.$$
$$D(C) \equiv C^d \text{ (mod } n\text{), for a ciphertext } C.$$

RSA encryption key is a pair of positive integers (e, n), and decryption key is a pair of positive integers (d, n) [19]. The first step is to compute n as the product of two very large random prime numbers p, and q, such that:

$$n = p \cdot q.$$

Although n will be made public, the factors p and q can be unknown to public due to the great difficulty in factoring n. Hence, this also hides the way d can be derived from e [12], [19], [22].

In the second step, d, a large random prime number that is relative to $(p – 1) \cdot (q – 1)$, or greater than max(p,q) is picked that satisfies [19]:

$$\gcd(d, (p – 1) \cdot (q – 1)) = 1$$

(gcd means greatest common divisor). It is important that d is substantially large so that a cryptanalyst cannot find it easily by direct search [19].

Finally, the integer e is computed from p, q, and d as the multiplicative inverse of d, modulo $(p – 1) \cdot (q – 1)$, such that [19]:

$$e \cdot d \equiv 1 \text{ (mod } (p – 1) \cdot (q – 1)).$$

Computing M^e (mod n) requires at most $2 \cdot \log_2(e)$ multiplications and $2 \cdot \log_2(e)$ divisions using a procedure called "exponentiation by repeated squaring and multiplication" [19].

Basically, RSA initially required each user to privately choose two very large (100-digit) random numbers p and q, so that upon computation n yields at least a 200-digit integer. It would be better, if the two numbers selected are not close to each other. So, the numbers should be so large that it is not computationally practical for anyone to factor $n = p \cdot q$, to crack the key [19]. Rivest et.al knew from the beginning that factoring n would enable attackers to break RSA. They knew Pollard's algorithm could factor a number n in $O(n^{1/4})$ time, and an algorithm by Schroeppel could factor n in even faster time [19].

RSA was adopted by NIST as ANS.X9.31 and later as PKCS#1. Both of these standards approved in FIPS186-4, subject to some additional requirements [16]. FIPS-186-4 specifies methods for generating RSA key pairs for several key sizes for ANSX9.31 and PKCS#1 implementations. RSA cipher uses only one round of operation and 1024 bits to 4096 bits key sizes [16].

H. Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is detailed in ANS X9.62, and is approved by FIPS186-4 with some additional requirements [16]. ECDSA produces digital signatures that are twice the length of the 160 bits key size [16].

I. Key Establishment, Agreement and Establishment Schemes

Key-establishment schemes are used to set up keys to be used between communicating parties. There are two types of key-establishment schemes - key transport and key agreement. Best key establishment schemes that use public-key algorithms are adopted in SP800-56 [23] from ANSX9.42 and ANSX9.63. ANSX9.42 details key agreement schemes and ANSX9.63 details both key agreement and key transport schemes [12].

Discrete Log Key agreement schemes use Finite-Field calculations. SP800-56 recommends eight key agreement schemes that are based on the complexity of the discrete logarithm problem and that use finite-field arithmetic for use [23]. Each scheme uses key pairs depending as per communication requirements [12], [18].

Discrete Log Key agreement schemes use Elliptic-Curve calculations. SP800-56 recommends seven key agreement schemes based on the complexity of the discrete logarithm problem and that use elliptic-curve arithmetic for use [18], [23]. Each scheme uses key pairs depending as per communication requirements [12]. Key establishment protocols also use key establishment schemes to specify the steps to establish a key. They also specify message flow and format. Thus, key establishment protocols must be carefully designed to prevent leak of secret information to a threat agent [18]. If given enough time and computer power to perform certain computations on the value of the secret or private key in use, then an attacker may be able to deduce the key from observed fluctuations using cryptanalysis techniques [12], [18].

Table 3 summarizes the current estimates for the maximum security strengths that the recommended symmetric and asymmetric cryptographic algorithms provide, with keys of a specific length [12]. Column 1 shows estimated maximum security strengths (in bits). Column 2 shows the symmetric-key algorithms that provide the security strength indicated in column 1 [12]. Column 3 shows the minimum size of the parameters associated with the standards that use finite-field cryptography (FFC). DSA is defined in FIPS186 for digital signatures and Diffie-Hellman (DH) is defined in SP800-56A [24]. L is the size of the public key and N is the size of the private key [12]. Column 4 indicates the value for k (the size of the modulus n) for algorithms based on integer-factorization cryptography (IFC). The predominant algorithm of this type is the RSA algorithm. RSA detailed in [FIPS186] for digital signatures, and in [SP800-56B] for key establishment. The value of k is the key size [12]. Column 5 shows the range of f (the size of n, where n is the order of the base point G) for algorithms based on elliptic-curve cryptography (ECC). ECC is specified for digital signatures in ANSX9.62 and adopted in FIPS186. For key establishment it is detailed in SP800-56A. The value of f is the key size [12]. The 192-bit and 256-bit key strengths identified for the FFC and IFC algorithms (in red) are not recommended because of interoperability and efficiency problems [12].

TABLE 3. COMPARABLE SECURITY STRENGTH OF BEST SYMMETRIC KEY AND ASYMMETRIC KEY ALGORITHMS [12]

Security Strength	**Symmetric key algorithms**	**FFC (DSA, D-H)**	**IFC (RSA)**	**ECC (ECDSA)**
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

III. SECURITY GAPS DUE TO QUANTUM COMPUTING

Post quantum cryptography concerns are not new, as Diffie and Hellman pointed these out in their paper – “New Directions in Cryptography” in 1976 [25]. Peter Shor’s paper in 1999 titled “Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”, proved that Feynman’s predicted quantum computer [3], [4], [5] was not very far [26], [27]. In his paper, he showed how randomized algorithms, factoring of integers and finding discrete logarithms, that were considered difficult for classical computers, the basis on which they were selected as cryptosystems, are easily broken in polynomial-time using a hypothetical quantum computer [27]. Hence, our current cryptosystems need a re-look to discover better algorithms for security in order to protect against cyber-attacks in the quantum computer era.

Supporting the aforementioned mentioned foundations is PQCrypto [28], an organization formed in early 2000’s by Deneiel J. Bernstein and Tanga Lange, have been encouraging post-quantum cryptography research and publications. Their website “Post quantum cryptography” contains numerous latest research publications on the issue, and they still believe that more research is required on the issue [28].

According to the NISTIR 8105 report, a “Report on Post-Quantum Cryptography”, current best and recommended cryptography algorithms Advanced Encryption Standard 256 (AES-256), Secure Hash Algorithm 3 (SHA-256), Secure Hash Algorithm 256 (SHA-256), Rivest Shamir Adleman (RSA), Elliptic Curve Digital Signature Algorithm (ECDSA), Elliptic-Curve Diffie–Hellman (ECDH), and Digital Signature Algorithm (DSA) which uses Finite Field Cryptography will not be secure for digital communications in post-quantum computing era [11].

This is because quantum computer by their quantum mechanical nature can proficiently solve these algorithms and any other BQP (bounded error, quantum, and polynomial time) problems (Fig.4) [29].

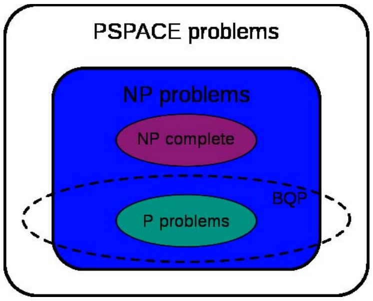


Figure 4. Problem Theory [29]

For some problems, quantum computers offer a polynomial speedup. Quantum computers will be able to solve BQP problems such as factorization and discrete logarithms operations in super-polynomial speed using

Shor's algorithm. Such an achievement is impossible in classical computers [30].

Integer factorization means the breakdown of a composite number into a product of smaller integers. If the final integers are restricted to prime numbers, the process is called prime factorization. So in integer factorization algorithm, given an n-bit integer, computer has to find the prime factorization. There are no efficient classical integer factorization algorithms. The general number field sieve which solves in a time $2^{\tilde{O}(n^{1/3})}$ is the fastest known classical algorithm for integer factorization. The best upper bound on the classical complexity of factoring is $O(2^{n/3+o(1)})$ [31].

In quantum computer, integer factorization is performed in super-polynomial speed. Peter Shor's quantum algorithm achieves this in $\tilde{O}(n^3)$ time [27]. Shor's factoring algorithm breaks RSA public-key encryption, while its related quantum algorithms for discrete logarithms break the DSA and ECDSA digital signature schemes, and the Diffie-Hellman key-exchange protocol. There also exists a quantum algorithm which is faster than Shor's for factoring "semi primes [32]. In the heart of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, and solved using the quantum Fourier transform [30]. Many cryptographic protocols are based on the difficulty of factoring large composite integers or a related problem—for example, the RSA problem.

Likewise, in Discrete-log algorithm, when given three n-bit numbers a, b, and N, where $b = a^s \bmod N$ for some s, finds s. Following Shor, this can be achieved on a quantum computer in poly(n) time [27]. The fastest known classical algorithm requires time super-polynomial in n. By similar techniques to those in [27], quantum computers can solve the discrete logarithm problem on elliptic curves, thereby breaking elliptic curve cryptography [33]. The super-polynomial quantum speedup has also been extended to the discrete logarithm problem on semi-groups [30], [34].

Likewise, Grover's algorithm in quantum computer can be applied to break a AES symmetric key algorithm by brute force in a time of about $2^{n/2}$ invocations of its underlying bits, compared with roughly 2^n in the classical computers [35]. So, symmetric key lengths are in effect halved, giving AES-256 the same level of security against an attack using Grover's algorithm that AES-128 has against brute-force search in classical computers. As of 2013, cryptanalysis attacks such as biclique attack and related-key attack that are computationally faster than brute force attack have been published for AES, but none tested computationally feasible [20], [21]. Grover's algorithm can also be used to obtain a quadratic speed-up over a brute-force search for NP-complete class of problems.

Current strong asymmetric / public-key cryptographic systems include RSA (Rivest-Shamir-Adleman) algorithm, elliptic curve algorithms such as ECDSA (EC-Digital Signature Algorithm) and ECDH (EC-Diffie-Hellman), and Finite Field algorithm such as DSA. They use either integer factorization or discrete log problem as their mathematical base [11] for digital signature and key exchange purposes, and hence also insecure for use in quantum era.

RSA is also based on the factoring problem – factoring the product of two large prime numbers [19], [22]. Cryptanalysis technique such general number field sieves for classical computers and Shor's algorithm for quantum computers leaves RSA-based public-key cryptography in a sorry state of security. As it is, a 768-bit RSA key has already been broken using cryptanalysis in classical computers [12], [22].

Elliptic Curve Digital Signature Algorithm (ECDSA) offers a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography. As with elliptic-curve cryptography in general, the bit size of the public key believed to be needed for ECDSA is about twice the size of the security level, in bits. For example, at a security level of 80 bits (meaning an attacker requires the equivalent of about 280 operations to find the private key) the size of an ECDSA public key would be 160 bits, whereas the size of a DSA public key is at least 1024 bits. On the other hand, the signature size is the same for both DSA and ECDSA: 4t bits, where t is the security level measured in bits, that is, about 320 bits for a security level of 80 bits [12],[36].

Elliptic Curve Diffie–Hellman (ECDH) is an anonymous key agreement protocol that allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key which can then be used to encrypt subsequent communications using a symmetric key cipher. It is a variant of the Diffie-Hellman protocol using elliptic curve cryptography [12], [37].

Table 4 shows a summary of common cryptographic algorithms that are under threat by quantum computers, because of their ability to solve BQP problems proficiently. This ability empowers quantum computers to decrypt many of the cryptographic systems in use today.

TABLE 4. COMMON CRYPTOGRAPHIC ALGORITHMS UNDER THREAT [11]

Cryptographic Purpose	Cryptographic Algorithm	Type	Impact from Quantum Computer
Encryption	AES-128	Symmetric Key	Larger key sizes needed
Hash Function	SHA-256, SHA-3		Larger output needed
Signatures, Key establishment	RSA	Public key	No longer secure
Signatures, Key exchange	ECDSA, ECDH (Elliptic Curve Cryptography)	Public key	No longer secure
Signatures, Key exchange	DSA (Finite Field Cryptography)	Public key	No longer secure

D-Wave X2 is quantum computer is now operational and Personal Quantum Computers (PQCs) may be developed any-time soon for commercial sale. To prevent hackers from having a field day compromising systems at unthinkable scale in the future, better cryptographic defenses have to be designed for post-quantum use. National Institute of Standards and Technology (NIST) has already begun the Post-Quantum Crypto Project in which it plans to standardize post-quantum cryptography [10]. So far, NIST has released a draft call-for-proposal document outlining submission requirements and evaluation criteria for post-quantum public key cryptography standards [38]. Soon it will begin accepting proposals from researchers for quantum-resistant public key encryption, digital signature, and key exchange algorithms. The deadline for submission is November 2017 [39], after which all proposals will undergo intense public scrutiny. Finally, NIST will select at least one algorithm for standardization [10].

IV. RECOMMENDATIONS

The following solutions will ensure development of quantum resistant cryptography for use in classical computers that can help prevent attacks by quantum computers and related technologies.

1. Use current unsolved-problems in mathematics as the mathematical base for the cryptographic algorithm. The chosen problem should be difficult enough even for a quantum computer to solve. Current unsolved problems in mathematics are listed in https://en.wikipedia.org/wiki/List_of_unsolved_problems_in_mathematics. The list includes Hilbert's problems, Landau's problems, Taniyama's problems, Thurston's problems, Smale's problems, Millennium prize problems (P vs NP, Hodge conjecture, Riemann hypothesis, Yang-Mills existence and mass gaps, Navier-Stokes existence an smoothness, Birch and Swinnerton-Dyer conjecture), and other unsolved problems in – algebra, algebraic geometry, analysis, combinatorics, discrete geometry, Euclidean geometry, dynamical systems, graph theory, model theory, and number theory [40].
2. Any algorithm that has been broken in classical computers so far has to be made obsolete for use in post-quantum era. If for example, AES-128 has been broken in classical computing attack then there is no use strengthening it with larger key size of 1024, as quantum computing attack will be able to defeat it anyway.
3. The chosen algorithms should not use finite field, integer factorization, or discrete log problem as their mathematical base as they can be efficiently solved using quantum computing capabilities.
4. If the keys are used, they should be substantially large integers (more than 300-digit) prime numbers.
5. Develop cryptographic algorithms that cannot be broken by Shor's and Gover's algorithms on quantum computers.
6. Develop cryptographic algorithms that cannot be broken by any of the powerful algorithms patented by D-Wave in the development of D-Wave X2 computer.
7. McEliece and Lattice-based cryptosystems that are also currently not known to be broken by quantum computers can be also used for now.
8. In the future, pure quantum cryptography that use quantum physics characteristics such as photons and electrons can be designed for quantum computer use only.

V. CONCLUSIONS

There is no doubt that we live in a time when top-most Cybersecurity implementations are vital in technologies we use for our daily communications service needs. The rise of quantum computing technologies such as D-Wave quantum computer will pose security threat to the current cryptographic defenses. Hence, there is a vital need to develop better cryptographic systems that can provide post-quantum protection in classical computers, and can interoperate with conventional networks and protocols. The recommendations provided as solutions can be used to devise better cryptographic algorithms for future use.

REFERENCES

[1] EY, "Creating trust in the digital world - EY's Global Information Security Survey 2015," EYGM Limited, 2015.

[2] Norse, Oct 2016. [Online]. Available: http://map.norsecorp.com/#/.

[3] R. P. Feynman, "Quantum Mechanical Computer," *Optics News,* pp. 11-20.

[4] R. P. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics,* vol. 21, no. 6/7, pp. 467-488, 1982.

[5] R. P. Feynman, "Space-Time Approach to Non-Relativistic Quantum Mechanics," *Reviews of Modern Physics,* vol. 20, no. 2, pp. 367-387, April 1948.

[6] D-Wave Systems Inc, "The D-Wave Quantum Computer," 2015. [Online]. Available: http://www.dwavesys.com/sites/default/files/D-Wave-brochure-Mar2016B.pdf. [Accessed 1 October 2016].

[7] top500, "Top 500 List - June 2016," [Online]. Available: https://www.top500.org/list/2016/06/ .

[8] Wikipedia, "Exynos," Oct 2016. [Online]. Available: https://en.wikipedia.org/wiki/Exynos. [Accessed 1 October 2016].

[9] Intel, "7th Generation Intel Core i7 Procesors," 10 October 2016. [Online]. Available: http://www.intel.com/content/www/us/en/processors/core/core-i7-processor.html. [Accessed 18 October 2016].

[10] NIST, "Post-Quantum Crypto Project," National Institute of Standards and Technology, August 2016. [Online]. Available: http://csrc.nist.gov/groups/ST/post-quantum-crypto/index.html. [Accessed 20 September 2016].

[11] NISTIR 8105, "Report on Post-Quantum Cryptography," 2016.

[12] NIST-SP800-57, "Recommendation for Key Management, Part 1: General, Rev.4," National Institute of Standards and Technology, Gaithersburg, 2016.

[13] Wikipedia, "Cryptographic hash function," 1 October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function . [Accessed 10 October 2016].

[14] FIPS180-4, "Secure Hash Standards (SHS)," National Institute of Standards and Technology, Gaithersburg, 2012.

[15] FIPS202, "SHA-3 Standard: Permutation-based hash and Extendable Output Functions," National Institute of Standards and Technology, Gaithersburg, 2015.

[16] FIPS186-4, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, Gaithersburg, 2013.

[17] FIPS-198-1, "The Keyed-Hash Message Authentication Code (HMAC)," National Institute of Standards and Technology, Gaithersburg, 2008.

[18] Wikipedia, "Cryptography," Wikipedia, 18 October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Cryptography. [Accessed 19 October 2016].

[19] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM,* vol. 21, no. 2, pp. 120-126, Feb 1978.

[20] FIPS197, "Announcing the Advanced Encryption Standard (AES)," Institute of Standards and Technology, Gaithersburg, 2001.

[21] ISO/IEC 18033-3, "Information technology - Security techniques - Encryption algorithms - Part3: Block Ciphers," 2010.

[22] Wikipedia, "RSA (cryptosystem)," Oct 2016. [Online]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem). [Accessed 10 October 2016].

[23] N. SP800-56, "Recommendation on Key Establishment Schemes".

[24] NIST800-56A (Rev.2), "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography," National Institute of Standards and Technology, Gaithersburg, 2013.

[25] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory,* vol. IT.22, no. 6, pp. 644-653, 1976.

[26] E. Rieffel, "An Introduction to Quantum Computing for Non-Physicists," Palo Alto, 2000.

[27] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Review, Society for Industrial and Applied Mathematics,* vol. 41, no. 2, pp. 303-332, 1999.

[28] PQCrypto, "Post-quantum cryptography," PQCrypto, 28 August 2016. [Online]. Available: https://pqcrypto.org/. [Accessed 27 September 2016].

[29] Wikipedia, "Quantum Computing," October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Quantum_computing. [Accessed 20 September 2016].

[30] S. Jordan, "Quantum Algorithm Zoo," The National Institute of Standards and Technology (NIST), 22 April 2011. [Online]. Available: http://math.nist.gov/quantum/zoo/. [Accessed 12 October 2016].

[31] M. O. Rubinstein, "The Distribution of solutions to XY = N (MOD A) with an application to Factoring Integers," *Integers,* no. 13, 2013.

[32] F. Grosshans, T. Lawson, F. Morain and B. Smith, "Factoring Safe Semiprimes with a Single Quantum Query," *arxiv:1511.04385v2,* 2016.

[33] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptive curves," *arxiv:quant-ph/0301141v2,* 2004.

[34] A. M. Childs and G. Ivayos, "Quantum computation of discrete logarithms in semigroups," *arxiv:1310.6238v2,* no. 11, 2013.

[35] C. H. Bennet, E. Bernstein, G. Brassard and U. Vazirani, "The Strengths and Weakenesses of Quantum Computation," *SIAM Journal of Computing,* vol. 26, no. 5, pp. 1510-1523, 1997.

[36] Wikipedia, "Elliptic Curve Digital Signature Algorithm," October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm. [Accessed 10 October 2016].

[37] Wikipedia, "Elliptic Curve Diffie-Hellman," October 2016. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic_curve_Diffie%E2%80%93Hellman. [Accessed 10 October 2016].

[38] NIST, "Proposed Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process," August 2016. [Online]. Available: http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-draft-aug-2016.pdf. [Accessed 20 September 2016].

[39] NIST, "Post-Quantum Crypto Project - Workshops," National Institute of Standards and Technology, August 2016. [Online]. Available: http://csrc.nist.gov/groups/ST/post-quantum-crypto/workshops.html. [Accessed 20 September 2016].

[40] Wikipedia, "List of Unsolved problems in Mathematics," Wikipedia, October 2016. [Online]. Available: https://en.wikipedia.org/wiki/List_of_unsolved_problems_in_mathematics. [Accessed 20 October 2016].