sensors

MDPI

Article

A Hash-Based Quantum-Resistant Chameleon Signature Scheme

P. Thanalakshmi [1], R. Anitha [1], N. Anbazhagan [2], Woong Cho [3], Gyanendra Prasad Joshi [4] and Eunmok Yang [5,*]

[1] Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore 641004, India; ptl.amcs@psgtech.ac.in (P.T); ani.amcs@psgtech.ac.in (R.A.)
[2] Department of Mathematics, Alagappa University, Karaikudi 630004, India; anbazhagann@alagappauniversity.ac.in
[3] Department of Software Convergence, Daegu Catholic University, Gyeongsan 38430, Korea; wcho@cu.ac.kr
[4] Department of Computer Science and Engineering, Sejong University, Seoul 05006, Korea; joshi@sejong.ac.kr
[5] Department of Information Security, Cryptology and Mathematics, Kookmin University, Seoul 02707, Korea
* Correspondence: emyang@kongju.ac.kr

Abstract: As a standard digital signature may be verified by anybody, it is unsuitable for personal or economically sensitive applications. The chameleon signature system was presented by Krawczyk and Rabin as a solution to this problem. It is based on a hash then sign model. The chameleon hash function enables the trapdoor information holder to compute a message digest collision. The holder of a chameleon signature is the recipient of a chameleon signature. He could compute collision on the hash value using the trapdoor information. This keeps the recipient from disclosing his conviction to a third party and ensures the privacy of the signature. The majority of the extant chameleon signature methods are built on the computationally infeasible number theory problems, like integer factorization and discrete log. Unfortunately, the construction of quantum computers would be rendered insecure to those schemes. This creates a solid requirement for construct chameleon signatures for the quantum world. Hence, this paper proposes a novel quantum secure chameleon signature scheme based on hash functions. As a hash-based cryptosystem is an essential candidate of a post-quantum cryptosystem, the proposed hash-based chameleon signature scheme would be a promising alternative to the number of theoretic-based methods. Furthermore, the proposed method is key exposure-free and satisfies the security requirements such as semantic security, non-transferability, and unforgeability.

Keywords: digital signature; chameleon signature; hash-based cryptography; homomorphic hash function; Preimage Resistance; key exposure free; random oracle model

Citation: Thanalakshmi, P.; Anitha, R.; Anbazhagan, N.; Cho, W.; Joshi, G.P.; Yang, E. A Hash-Based Quantum-Resistant Chameleon Signature Scheme. *Sensors* **2021**, *21*, 8417. https://doi.org/10.3390/s21248417

Academic Editors: Athanasios V. Vasilakos and Alexios Mylonas

Received: 1 October 2021
Accepted: 10 December 2021
Published: 16 December 2021

1. Introduction

An ordinary digital signature is not suitable for all applications that are personally or commercially sensitive. Consider the scenario where a software company issues a license of the software to the customer. When the company embeds an ordinary signature into its product to assure the customer Alice that the vendor provides the software, Alice can make several copies of the software and sell them at a lower price. She can prove that the software is from the vendor by presenting the vendor's signature on the product. A privacy-protecting signature can be embedded in the product by protecting the company's reputation and assuring the customer of having quality software. Though undeniable signature [1] proposed by Chaum et al. provides controlled dissemination of signature, a group of verifiers can verify it simultaneously, without telling the prover that he is proving the signature to too many people. This is done by setting the challenge collectively so that no subset of the verifiers could set the challenge on their own. To overcome the above-mentioned limitations of undeniable signature, Krawczyk & Rabin introduced chameleon signature [2], which prevents the dissemination of signed information by the

recipient of the signature. It also preserves the non-repudiation property in the event of legal disputes.

Though undeniable signature and chameleon signature both provide non-transferability and non-repudiation, the former requires an interactive protocol between the signer and the verifier for the purpose of verifying the signature, which is based on zero-knowledge proof. It is quite natural that such additional properties and techniques increase the complexity of undeniable signature relative to ordinary signature in terms of computational and communication costs. As the chameleon signature scheme is non-interactive, it does not require the zero-knowledge paradigm, and hence its computational complexity is less compared to the undeniable signature scheme. Like an ordinary digital signature scheme, the chameleon signature scheme is constructed with the basic approach of the hash-and-sign paradigm. First, the message digest is computed using a special hash function called chameleon hash function, which uses the public key of the recipient of the signature. Like a cryptographic hash function, it provides collision resistance except for the trapdoor holder (recipient) of the public key. As a result, finding collisions to chameleon hash value is computationally infeasible even for the signer who produces the hash value, but it is feasible for the recipient to compute collisions using his trapdoor key. When a recipient receives a chameleon signature on a message, through collision finding, the recipient is able to produce different messages for the same signing value as the original one. Thus, the underlying chameleon hash function allows the recipient to forge the signature generated by the signer. The signatures generated by the signer and by the recipient are indistinguishable. This ability of the recipient to forge an indistinguishable signature prevents him from revealing the validity of the original signature made by the signer to a third party. Thus, the chameleon hash function is non-transferable. However, in the case of disputes, the signer can easily prove that the contested signature by the recipient is a forgery by producing a message–signature pair that has the same hash value as that of the contested message–signature pair. As the signer cannot compute a collision, the message–signature pair produced by the signer is taken as original, while the contested signature is considered forgery. The early designs of chameleon signature schemes are key-exposure, i.e., collisions that reveal the recipient's secret key. Because the key is exposed, the recipient is unable to use the signature scheme. Calculating hash collisions, putting the principle of non-transferability to the test. As a result, the major key exposure concern, as well as its related issues such as revocation and redistribution of keys, must be addressed. Henceforth, a key-exposure-free chameleon hash is proposed in a graph setting. It uses a double trapdoor mechanism. The proposed chameleon hash system is used to create a chameleon signature scheme. The non-transferability of the signature thus produced is simply achieved from the underlying chameleon hash.

2. Related Work

Krawczyk & Rabin proposed the first chameleon signature scheme in [2]. It suffers from a key exposure problem which means that, when the recipient does forgery through collision computation, the signer can easily retrieve the recipient's secret key. This creates a strong disincentive for the recipient to forge chameleon signatures which partially undermines the concept of non-transferability. Zhang et al. presented chameleon hash schemes in an ID-based setting in [3]. The proposed two schemes from bilinear pairings are not key-exposure-free. Ateniese & de Medeiros in [4] identified that the problem of key exposure on forgeries threatened the claim of non-transferability and provided a solution to the problem through an identity-based chameleon hash function. In their scheme, the signature forgery by the recipient results in the recovery of the recipient's trapdoor information associated with that single transaction by the signer. However, in other transactions, the signer could not be able deny the signature on any message. Thus, the authors provide a solution but do not solve the problem of key exposure completely. To overcome this drawback, Chen et al. proposed the first complete key-exposure free chameleon hash function in the gap Diffie–Hellman group with bilinear pairings [5].

In [4], Ateniese & de Medeiros introduced three nonpairing based key exposure-free systems, two of which are based on Integer Factorisation Problem (IFP) and one on Diffie–Hellman and the Discrete Logarithm Problem (DLP). Gao et al. developed a factoring-based chameleon hash function [6] and demonstrated its security by using a modified Rabin signature technique. Later, Gao et al. introduced a DLP-based key-exposure free chameleon hash function [6]. They modified the basic construction of the chameleon hash method [2] to allow a multi-trapdoor by considering the hash algorithm as a one-round protocol that runs between the signer and the specified recipient. This architecture differs slightly from Krawczyk & Rabin's original concept of chameleon hash. Later, Chen et al. suggested a DLP-based key-exposure-free chameleon hash and signature system that did not need the gap Diffie–Hellman groups [7]. Pan et al. proposed a family of chameleon hash functions and strongly unforgeable one-time signature schemes based on DLP over inner automorphism groups [8].

Integer factorization and discrete logarithm problems provide a secure basis for the above schemes and are threatened by quantum computers by Shor's algorithm [9]. Therefore, it is necessary to come up with alternative schemes that resist quantum computer attacks. Although quantum computers are currently in their infancy [10,11], the ongoing development of quantum computers and their theoretical ability to compromise modern cryptographic schemes has prompted the development of post-quantum cryptographic schemes. A cryptographic technique connected with a computational problem that cannot be solved in polynomial time on a quantum computer is currently referred to as post-quantum security. This includes the fact that addressing the problem on a traditional computer is impossible. Because lattice-based signatures are thought to be secure against quantum computers, they have become fascinating alternatives to the current techniques, such as RSA, ECDSA, and others. Several lattice-based cryptographic signature techniques have been proposed since Ajtai's original work [12]. In hard random lattices, Xie et al. suggested a homomorphic chameleon function based on the Small Integer Solution (SIS) problem [13].

Other promising post-quantum signature schemes are hash-based signature schemes. Hash functions are sufficient to have an efficient and a secure transmission of data. As hash functions are one-way, hash-based signatures provide certain advantages over digital signatures based on trapdoor functions. The following are the advantages of hash-based signatures: they require less security assumptions than number-theoretic signature schemes, they are very efficient, and their security is well understood. The generic attacks on hash function and pseudorandom generator determine the security parameter. The minimum security parameter recommended is determined by the security level of preimage and collision resistance of the hash function. When classical computers are used, then finding a preimage of the hash function $H : \{0,1\}^{2k} \rightarrow \{0,1\}^{k}$ by exhaustive search costs $2^{k/2}$ and finding a collision by birthday attack costs $2^{k/2}$. However, when quantum computers are used, the Grover algorithm [14] requires $2^{k/3}$ evaluations of the hash function to find a collision and requires $2^{k/3}$ searches to find a preimage of H. For a pseudorandom generator $PG : \{0,1\}^{k} \rightarrow \{0,1\}^{2k}$, the attacker tries to find the patterns in the inputs example by guessing the PG seed that ends up at a target value after a number of rounds of the chaining function. All generic attacks again cost 2^{k} in pre-quantum and $2^{k/2}$ in post-quantum.

Chen [15] proposed a PDP protocol which consists of an algebraic signature that uses a hash function with homomorphic property. It is shown that the homomorphic hash function enables fast and efficient retrieval of content and provides provable data possession and data integrity protection in cloud storage. Hence, this paper proposes a simple construction of a chameleon hash scheme under graph setting with minimal requirements such as homomorphic hash function and homomorphic pseudorandom generator and without complex algebraic computation. The security of the chameleon hash scheme relies on the Preimage Resistance (PR) of the hash function. The chameleon signature scheme constructed with the proposed chameleon hash scheme easily achieves the non-transferability

property. In addition, the method obtains the unforgeability property directly from the unforgeability of the underlying signature scheme used.

The paper is organized as below: Section 3 contains a brief description of basic concepts of graph theory, the outline of a chameleon hash and signature scheme, and its security model. Section 4 proposes an efficient hash-based chameleon hash and signature scheme with DAG, and Section 5 presents an algorithm for the construction of a DAG. Section 6 presents the security analysis of the proposed method, and Section 7 discusses the performance analysis and comparison. Finally, Section 8 concludes the paper.

3. Preliminaries

In this section, few graph preliminaries [16] that are required for the proposed scheme and definitions and properties of chameleon hashing and signatures are briefed.

Definition 1. ***Directed graph:*** *It is an ordered pair $G = (V, E)$ where V is a finite set of elements called vertices and E is a set of ordered pairs of vertices called directed edges.*

Definition 2. ***Degree:*** *In a directed graph, indegree of a vertex v is the number of edges $(u, v) \in E$ which are pointing to the vertex v. Outdegree of a vertex v is the number of edges $(v, u) \in E$ which are coming from the vertex v. The sum of indegree and outdegree is the total degree of v.*

In a directed graph, the vertices with indegree zero are called source vertices and the vertices with outdegree zero are called sink vertices. The vertices which are neither sources nor sinks are called internal vertices.

Definition 3. ***Path:*** *In a directed graph, a path (v_0, v_l) is a sequence of vertices $(v_0, \ldots, v_l)$ such that $(v_{i-1}, v_i) \in E$, $\forall i = 1, \ldots, l$. Its length is the number of occurrences of edges in it.*

Definition 4. ***Cycle:*** *In a directed graph, a cycle is a closed path in which all the vertices are distinct except that $v_0 = v_l$.*

Definition 5. ***DAG:*** *A directed acyclic graph is a directed graph with no cycles.*

For simplicity, only DAGs which have a single source s with outdegree 1, a single sink t with indegree 1, and $n > 0$ internal vertices where each vertex that has a total degree of 3 is considered. Such graphs have only two types of internal vertices, namely expansion vertices and compression vertices. Expansion vertices are vertices with indegree 1 and outdegree 2, and compression vertices are those with indegree 2 and outdegree 1. A DAG G with n internal vertices where n is even will have $n/2$ expansion vertices and $n/2$ compression vertices. This particular type of DAG is considered throughout the paper.

Definition 6. ***Cut;*** *In a DAG $G = (V, E)$ with a source edge e_s and a sink edge e_t, a cut is defined as a nontrivial partition $W = (U, V \setminus U)$ of the vertices such that e_s is in the edge set of U and e_t is in the edge set of $V \setminus U$. A cut is represented by a single set of vertices U with the convention that $e_s \in U$.*

Definition 7. ***Cross edge:*** *A cross edge $e = (u, v)$ of a cut $W = (U, V \setminus U)$ is defined as an edge in which $u \in U$ and $v \in V \setminus U$. The set of edges crossing W is denoted as (W).*

Definition 8. ***Poset;*** *A set P with a reflexive, antisymmetric and transitive relation $\preceq$ defined on it is a partially ordered set or poset denoted by $(P, \preceq)$.*

Definition 9. ***Comparable:*** *Two elements a and b of a poset $(P, \preceq)$ are called comparable if either $a \preceq b$ or $b \preceq a$. When neither $a \preceq b$ nor $b \preceq a$, then the elements a and b are called incomparable.*

Definition 10. ***Antichain:*** *A subset $U \subseteq P$ is called an antichain if every pair of elements of U is incomparable. The maximal cardinality of antichains in P is called the width of a poset $(P, \preceq)$, and it is denoted by $w(P)$.*

Definition 11. ***Allowable set:*** *Let T be a set of edges in G. The set of directly computable nodes is defined as the set of nodes all of whose incoming arcs are in T. The set of computable nodes is defined recursively as the set of nodes computable from the union of T with the outgoing edges of the directly computable nodes. The set T is called an allowable set if it is verifiable and consistent. When T is allowable, the public key is obtained from the set of its computable nodes and the set either contains all the outgoing edges from a node or none from it.*

Definition 12. ***Minimal allowable set:*** *An allowable set is called a minimal allowable set if it is not verifiable when any edge is omitted from the set.*

Definition 13. ***Poset:*** *It is a set $(G^*, \preceq)$ that consists of minimal allowable sets of G with a partially ordered relation $\preceq$ defined on it. Two minimal allowable edge sets U and V of a poset are related as $U \preceq V$ if and only if the set of computable nodes of V is a subset of the set of computable nodes of U.*

Definition 14. ***Comparable:*** *Two minimal allowable sets in G are said to be compatible if neither of the sets of computable nodes is a subset of the other. Then, the corresponding sets of computable nodes are incomparable elements in the associated poset.*

3.1. Chameleon Hash Scheme

A chameleon hash function is a collision-resistant trapdoor hash function linked to a public/secret key combination (pk, sk). Anyone with access to the public key pk can easily compute the hash value for each input. Except for the trapdoor key sk holder, no one can efficiently compute a collision on any given input.
The following three polynomial-time algorithms define the formal definition of a chameleon hash scheme [7]:

Key Generation (1^k)**:** A probabilistic polynomial-time algorithm that generates a public/secret key pair (pk, sk) based on the security parameter k.
Hashing Computation: A probabilistic polynomial-time algorithm that computes and outputs the chameleon hash value $\mathcal{G}_{pk}(M, r)$ given a public key pk, a message M, and a random parameter r.
Collision Computation: A deterministic polynomial-time method that outputs r' such that $\mathcal{G}_{pk}(M, r) = z^{'} = \mathcal{G}_{pk}(M', r')$ when given the secret key sk, a message M, a random parameter r, and another message M' as inputs.

Furthermore, if r is taken from a uniform probability distribution, the distribution of r' is computationally indistinguishable from the uniform.

The following properties should be met by a secure chameleon hash technique [7]:

Collision resistance: Without knowing the trapdoor key sk, there is no efficient algorithm that, given a message M, an auxiliary random parameter r and another message M', outputs r' with a non-zero probability that $\mathcal{G}_{pk}(M, r) = z^{'} = \mathcal{G}_{pk}(M', r')$.
Semantic security: Let $E[X]$ denote the entropy of a random variable X, and $E[X|Y]$ denote the entropy of a random function Y of X given the value of the variable X. If the conditional entropy $E[M|z']$ of a message given its chameleon hash value z' equals the total entropy $E[M]$ of the message space, the system is semantically secure.
key-exposure freeness: If a recipient has never computed a collision on a chameleon hash $\mathcal{G}_{pk}(M, r)$, no adversary will be able to identify an efficient algorithm to compute a collision on that $\mathcal{G}_{pk}(M, r)$. This is true even if the adversary has access to the collision computation oracle for polynomially many queries on tuples (M_i, r_i) of his choosing, except for the challenge query.

3.2. Chameleon Signature Schemes

A message's chameleon signature is created by digitally signing the message's chameleon hash value. A chameleon signature scheme, as defined by Chen et al., consists of the techniques described in [7]:

Key Generation (1^k)**:** A probabilistic polynomial-time algorithm that outputs a public/secret key pair (pk_S, sk_S) for the signer and (pk_V, sk_V) for the verifier based on the security parameter k.
Sign: A probabilistic polynomial-time algorithm that outputs a signature σ on the chameleon hash value $z^{'}$ using the recipient's public key pk_V, the signer's secret key sk_S, a random string r, and a message M as inputs.
Verify: A deterministic polynomial-time procedure that outputs *Accept* if σ is valid else outputs *Reject* given the the recipient's public key pk_V, the signer's public key pk_S, the random string r, the message M, and the signature σ.
Denial protocol: A protocol between the signer and the judge that is not interactive. The signer submits a valid collision (M', r') to the judge when given a chameleon signature σ on a message M. If $M \neq M'$ and σ are both legitimate, the judge concludes that the signature σ on the message M is forged.

3.3. Security Requirements of Chameleon Signature Schemes

The following are the properties of a chameleon signature scheme: unforgeability, non-transferability, non-repudiation, and deniability:

Unforgeability: A valid chameleon signature cannot be generated by a third party. Furthermore, a recipient can only produce a forgery for a chameleon signature that was previously generated by the signer.
Non-transferability: Because the recipient can forge a signature, the recipient cannot persuade a third party that the signer generated a signature on a specific message.
Non-repudiation: The signer is not allowed to deny legitimate signature claims.
Deniability: By providing a collision to the chameleon hash value, the signer can deny a forgery of the signature.

4. Proposed Hash-Based Chameleon Hashing and Signature Scheme

In this section, first, a chameleon hash function is constructed, and then a signature scheme is designed using the proposed hash function.

4.1. Construction of a Chameleon Hash Scheme

The proposed chameleon hash scheme consists of the following polynomial-time algorithms:

Key Generation (1^k)**:** Let G be a publicly known DAG. Associate a homomorphic hash function $H : \mathbb{F}_2^k \times \mathbb{F}_2^k \rightarrow \mathbb{F}_2^k$ in the compression vertices and a homomorphic pseudorandom generator $PG : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^k \times \mathbb{F}_2^k$ in the expansion vertices of G. The public parameters are G, H and PG with respect to the security parameter κ. The algorithm randomly chooses $sk \in \mathbb{F}_2^k$ and computes $pk = Ext_{\{e_t\}}[sk : G(e_s)]$. It chooses a cut W in G. Let (W) be the cross edges of the cut W with $|W| = l$. It chooses a labeling $r = (r_1, r_2, \ldots, r_l)$ for (W) where $r_i \in \mathbb{F}_2^k$ is the labeling of the i'th edge in (W) and computes $r^{'} = Ext_{\{e_t\}}[r : G(W)]$. It outputs $(pk, ((W), r^{'})$ as a public key, where pk is the permanent public key and $((W), r^{'})$ is the ephemeral public key which is changed periodically. The corresponding private key is (sk, r).
Hashing Computation: On input $(M, pk, (W), r^{'})$, the algorithm computes
$h^{'} = Ext_{\{e_t\}}[h(M) : G(W)]$
$\mathcal{G}_{pk}(M, r^{'}) = h^{'} \oplus r^{'} \oplus pk = z^{'}$
Outputs chameleon hash z', for the tuple (M, r').
Collision Computation For any valid hash value z', the collision computation algorithm computes a hash collision with the trapdoor key (sk, r) as follows: $z = h(M) \oplus r \oplus$

$Ext_{\{(W)\}}[sk : G(e_s)]$
$r^* = z \oplus h(M^*) \oplus Ext_{\{(W)\}}[sk : G(e_s)]$.
$r^{*'} = Ext_{\{e_t\}}[r^* : G(W)]$
The algorithm outputs a collision tuple $(M^*, r^{*'})$ for the chameleon hash value $z^{'}$.

4.2. Construction of the Chameleon Signature Scheme

A chameleon signature scheme is constructed by having the proposed chameleon hash function in a standard signature scheme of the same public key setting. The public key and the secret key of the proposed chameleon hash scheme are related in terms of a DAG. Hence, it is convenient to design a chameleon signature scheme by combining the proposed chameleon hash scheme and a graph-based signature scheme.

In the proposed signature scheme, each of the signer S and verifier V chooses a DAG and a secret value and labels the source edge of their DAG with the secret value and labels all the other edges by means of extension labeling from the source edge. The label obtained at the sink edge is their public key. The signer first computes the chameleon hash value for a message using the verifier's DAG G_V and his public key. Then, he computes all minimal allowable sets for the graph G_S and defines a partially ordered relation $\preceq$ on it. The associated poset of G_S is $(G_S^*, \preceq)$ with width $w(G_S^*)$. Then, with an efficient mapping function, the signer maps the chameleon hash value to a minimum allowable set in the antichain of width $w(G_S^*)$. Then, he outputs the values associated with the minimal allowable set with respect to his secret key as the signature for the chameleon hash value.

Signatures thus obtained are verifiable with respect to the signer's public key and are compatible which means that without inverting hash functions or pseudorandom functions, no signature can be computable from the signatures of different messages. Consequently, the essential requirements for the proposed signature scheme are maximal antichain of minimal allowable sets and an efficiently computable collision-resistant mapping from the hash space to the maximal antichain. In addition, for the chameleon signature scheme to be recipient-specific, the hash computation has to be done on the verifier's graph, and signature computation has to be done on the signer's graph.

The following algorithms describe the proposed Hash-Based Chameleon Signature (HBCS) scheme:

Key Generation (1^k)**:** The signer S and the verifier V selects DAG's G_i $(i = S, V)$. Let H be a homomorphic hash function and PG be a homomorphic pseudorandom generator associated with the compression vertices and expansion vertices of G_i $(i = S, V)$. The signer randomly chooses $sk_S \in \mathbb{F}_2^k$, computes $pk_S = Ext_{\{e_t\}}[sk_S : G_S(e_s)]$ and outputs the public key as $(pk_S, ((U), t^{'}))$ where pk_S is the permanent public key and $((U), t^{'})$ is the ephemeral public key. The corresponding private key is (sk_S, t). Similarly, the verifier on his graph G_V, computes and outputs the public key $(pk_V, ((W), r^{'}))$ where pk_V is the permanent public key and $((W), r^{'})$ is the ephemeral public key. The corresponding private key is (sk_V, r). Let $\mu : \{0,1\}^k \rightarrow G_S^*$ define a mapping from the chameleon hash space $\{0,1\}^k$ to the maximal antichain of G_S^*.

Sign (sk_S, M)**:** Let $M \in \{0,1\}^*$ be the message to be signed. With the verifier's secondary key $((W), r^{'})$, the signer computes the chameleon hash value $\mathcal{G}_{pk_V}(M, r^{'}) = z^{'}$ and maps $z^{'}$ to $\mu(z^{'})$ in the maximal antichain of G_S^*. Then, computes
$\lambda = Ext_{\{\mu(z^{'})\}}[sk_S : G_S(e_s)]$
The signature on M is $\sigma = (r^{'}, \lambda)$.

Verify (pk_S, M, λ)**:** The verifier computes
$\mathcal{G}_{pk_V}(M, r^{'}) = z^{'}$
Verifies $pk_S = Ext_{\{e_t\}}\left[\lambda : G_S(\mu(z^{'}))\right]$

Correctness

$$\begin{aligned} pk_S &= Ext_{\{e_t\}}[sk_S : G_S(e_s))] \\ &= Ext_{\{e_t\}}\left[Ext_{\{\mu(z')\}}[sk_S : G_S(e_s)] : G_S(\mu(z'))\right] \\ &= Ext_{\{e_t\}}\left[\lambda : G_S(\mu(z'))\right] \end{aligned}$$

Denial Protocol: In the event of a legal dispute between the recipient and the signer, i.e., when the recipient submits a message–signature tuple $(M^*, (r^{*'}, \lambda))$ to the judge J and claims that it was generated by the signer, the judge first applies the above signature verification process and determines whether $(r^{*'}, \lambda)$ is a proper signature on M^*. If the verification fails, the judge rejects the alleged signature. Otherwise, he requests that the signer denies or accepts the claim. If the signer wishes to accept the signature, he simply informs the judge. If the signer wishes to declare the signature to be a forgery, he must provide a collision tuple $(M, (r', \lambda))$ i.e., $\mathcal{G}_{pk_V}(M, r', \lambda) = \mathcal{G}_{pk_V}(M^*, r^{*'}, \lambda)$. It is worth noting that, if $(M^*, (r^{*'}, \lambda))$ is a forgery, the signer can always supply the original message–signature tuple $(M, (r', \lambda))$ that differs from $(M^*, (r^{*'}, \lambda))$. If $(M^*, (r^{*'}, \lambda))$ is valid, the signer cannot provide a collision on the hash value without knowing the secret key of the recipient. As providing collision is equivalent to finding the preimage of the hash function, which is shown in Theorem 1, the signer cannot provide collision and repudiate a valid signature. The inability of the signer in providing a collision on the chameleon hash value enables the judge to determine the signature is valid or forged.

5. Construction of DAG

Algorithm 1 is proposed to generate a DAG as explained in Section 3.

Algorithm 1 DAG Construction

let $edge[n][n]$ be the matrix denoting the edges and initialized to 0.
$edge[1][2] \leftarrow 1$
$edge[n-1][n] \leftarrow 1$
for $i \leftarrow 1$ to $n-2$ **do**
    $edge[i][i+2] \leftarrow 1$
end for
$k \leftarrow n/2$ ▷ Setting second edges for expansion vertices
if k is even **then**
    for $i \leftarrow 2$ to k **do**
        $edge[i][k+i-1] \leftarrow 1$
    end for
else if k is odd **then**
    for $i \leftarrow 2$ to k **do**
        if i is even **then**
            $edge[i][k+i] \leftarrow 1$
        else if i is odd **then**
            $edge[i][k+i-2] \leftarrow 1$
        end if
    end for
end if

DAG's G_V and G_S are constructed and shown in Figures 1 and 2, respectively.

In order to demonstrate the idea of the scheme, G_V and G_S are considered as the public graphs of verifier and signer, respectively. The hash function and pseudorandom generator are associated with the compression and expansion vertices of DAG's respectively. The signer chooses his secret key sk_S and computes the public key pk_S using G_S. Similarly, the verifier chooses sk_V and computes the public key pk_V using G_V. In addition, the verifier chooses a cut W in G_V and chooses random value r along the cross

edges $\{g, h, e, d, f\}$ of the cut and computes $r^{'} = Ext_{\{e_t\}}[r : G_V(W)]$. The verifier publishes $(pk_V, ((W), r^{'}))$ as public key. The signer computes the chameleon hash for $h(M)$ in G_V as $z^{'} = (Ext_{\{e_t\}}[h(M) : G_V(W)] \oplus r^{'} \oplus pk_V)$. He computes the associated poset G_S^* for G_S as shown in Figure 2. As the width of G_S^* is 5, the hash space consisting of at most 5 hash values can be mapped to the antichain of width 5. The elements in this antichain $\{e, a, h\}$, $\{b, c, h\}$, $\{c, d, e, f\}$, $\{g, a, f\}$ and $\{g, b, d\}$ are signature patterns. Suppose that the hash value $z^{'}$ is mapped to an element $\{g, a, f\}$, then the signature is the value of $r^{'}$ and the values of the edges $\{g, a, f\}$ with respect to the signer's secret key.

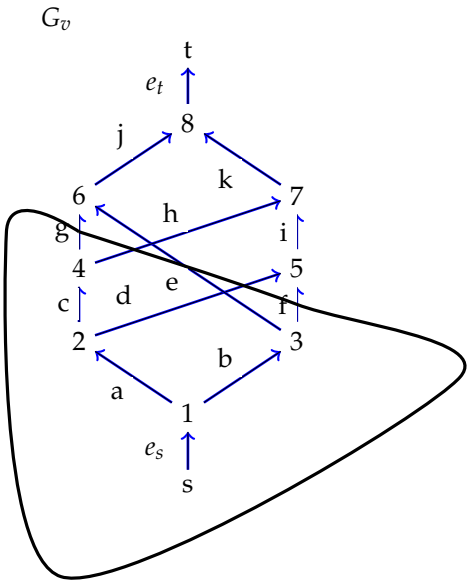

Figure 1. The locations of the studied lakes.

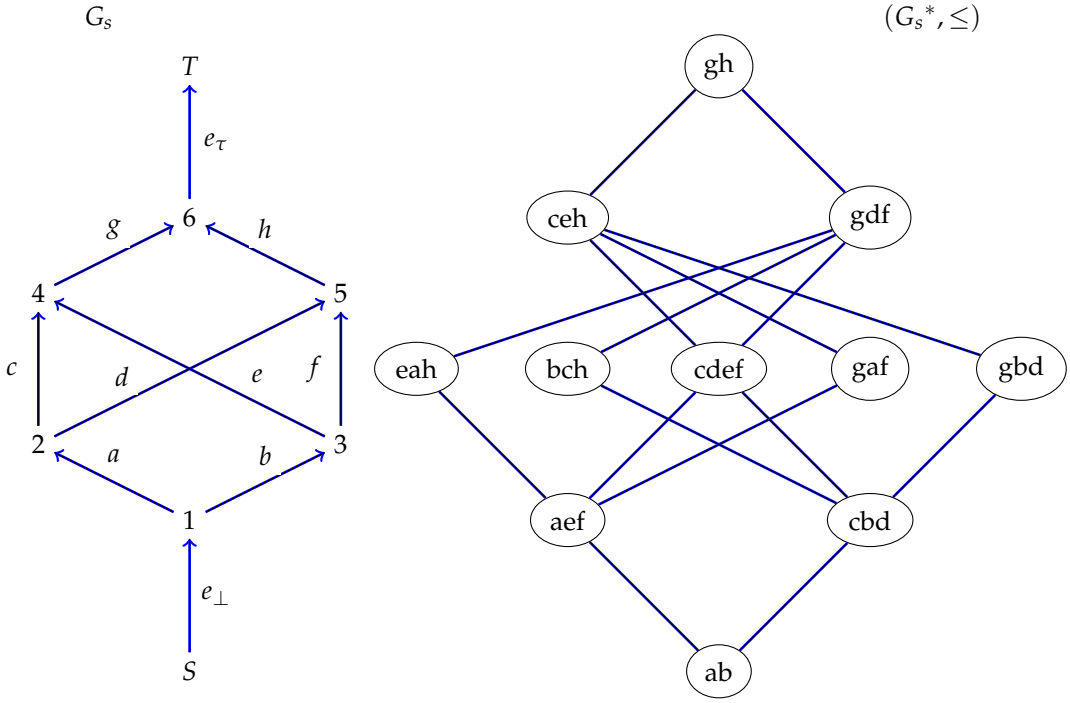

Figure 2. A DAG and its Poset.

6. Security Analysis

This section examines the proposed chameleon hash function's security properties. It also discusses the proposed chameleon signature scheme's security against forgery, transferability, repudiation, and undeniability.

6.1. Security Analysis of the Proposed Hash-Based Chameleon Hash Scheme

In this subsection, the three properties of the chameleon hash scheme—collision resistance, semantic security, and key-exposure freeness—are proved.

Theorem 1. *The proposed chameleon hash scheme is collision resistant provided the hash function is preimage resistant.*

Proof. Let A be an adversary against collision resistance of chameleon hash function. Assume that A outputs collision tuples (M, r') and $(M^*, r^{*'})$.
Therefore,
$\mathcal{G}_{pk_V}(M, r^{'}) = \mathcal{G}_{pk_V}(M^*, r^{*'}) = z^{'}$
$\Rightarrow h^{'} \oplus r^{'} \oplus pk_V = h^{*'} \oplus r^{*'} \oplus pk_V = z^{'}$
$\Rightarrow h^{'} \oplus r^{'} = h^{*'} \oplus r^{*'}$
$\Rightarrow h^{'} \oplus h^{*'} = r^{'} \oplus r^{*'}$
$\Rightarrow Ext_{\{e_t\}}[h(M) : G_V(W)] \oplus Ext_{\{e_t\}}[h(M^*) : G_V(W)] = r^{'} \oplus r^{*'}$
$\Rightarrow Ext_{\{e_t\}}[h(M) \oplus h(M^*) : G_V(W)] = r^{'} \oplus r^{*'}$
$\Rightarrow (h(M) \oplus h(M^*))$ is the trapdoor value of $(r^{'} \oplus r^{*'})$ along the cross edges of the cut W in G_V. Hence, by means of the consistent extension of $(h(M) \oplus h(M^*))$, the preimage of $(r^{'} \oplus r^{*'})$ under H is obtained. As finding the preimage of hash function is computationally infeasible, it can be concluded that, without the knowledge of the trapdoor, providing chameleon hash collision is hard. □

Theorem 2. *The proposed chameleon hash scheme is semantically secure.*

Proof. The computation of chameleon hash is $z^{'} = Ext_{\{e_t\}}[z : G_V(W)]$ which equals $Ext_{\{e_t\}}\left[\left(h(M) \oplus r \oplus Ext_{\{(W)\}}[sk : G(e_s)]\right) : G_V(W)\right]$, where the value of r is chosen completely independent of M. In addition, the equation $z = h(M) \oplus r \oplus Ext_{\{(W)\}}[sk : G(e_s)]$ implies that, for each fixed M, there is a one-to-one correspondence between the values of z and r.

Hence, $Pr(M|z^{'})$= $Pr(M|z)$ = $Pr(M|r)$ = $Pr(M)$. To prove the semantic security of the scheme, it is required to prove the conditional entropy $E[M|z^{'}] = E[M]$.

$$\begin{aligned} E[M|z^{'}] &= -\sum_{M}\sum_{z^{'}} Pr(M, z^{'}) log(Pr(M|z^{'})) \\ &= -\sum_{M}\sum_{z^{'}} Pr(M, z^{'}) log(Pr(M)) \\ &= -\sum_{M} log(Pr(M)) \sum_{z^{'}} Pr(M, z^{'}) \\ &= -\sum_{M} log(Pr(M)) Pr(M) = E[M]. \end{aligned}$$

Because the conditional entropy $E[(M|z^{'}]$ equals the total entropy $E[M]$, the chameleon hash value $z^{'}$ reveals no information about M. As a result, the chameleon hash technique suggested is semantically secure. □

Theorem 3. *The proposed chameleon hash technique does not require any key disclosure.*

Proof. Even if the adversary A has been given oracle access to collision computation and has been given polynomially many queries with the tuples $(M_i, r_i^{'})$, $i \geq 1$, of his choice, except for the challenge question, we assert that he cannot detect a collision on the hash value $z^{'} = \mathcal{G}_{pk}(M, r^{'})$ using an efficient technique. The adversary can use two sorts of attacks to find collision on $z^{'}$. He might try to recover the secret values of $r^{'}$ and pk along the cut's cross edges in the first attack, and he might try to discover collision on $z^{'}$ without recovering the secret keys in the second attack. The hash function's Preimage Resistance (PR) is the target of the first attack. The second approach is to identify a collision on the given challenge $z^{'}$ without knowing the private key, which is similar to cracking the graph-based one-time signature technique [16]. Because the graph-based one-time signature technique is safe against existential forgery in the standard model under a chosen message attack, it can be deduced that, even if the adversary has a polynomial number of signatures on messages, finding a collision on the hash $\mathcal{G}_{pk}(M, r^{'})$ is difficult. As a result, the suggested chameleon hash has no key exposure. □

6.2. Security Analysis of the Proposed Hash-Based Chameleon Signature Scheme

In this subsection, the properties of the chameleon signature scheme: unforgeability, non-transferability, non-repudiation, and deniability are proved.

Theorem 4. *Let $G(V, E)$ be a directed acyclic graph with $|V| = n$, PG be a homomorphic pseudorandom generator and H be a homomorphic hash function. Let A be an adversary who performs an existential forgery under a chosen message attack against the proposed HBCS scheme with success probability ϵ. Then, $\epsilon \leq (\alpha\epsilon_D + \epsilon_C + \epsilon_{PG} + \epsilon_H)n + \epsilon_{\mathcal{G}}$ where $\alpha \leq n$ is the average number of predecessors of a random vertex in the graph and ϵ_{PG}, ϵ_H,ϵ_C, ϵ_D, and $\epsilon_{\mathcal{G}}$ are, respectively, the success probabilities on inverting PG, inverting function H, finding collision on H, distinguishing random strings and pseudorandom strings apart, and finding collision on a chameleon hash function.*

Proof. As HBCS is based on the hash-and-sign approach, the adversary can produce forgery either by finding a valid collision on the chameleon hash or breaking the underlying signature. By Theorem 1, the signer cannot produce a valid collision on a chameleon hash value. The construction of the proposed underlying signature is based on the signature schemes proposed by Hevia et al. in [16] and Dod et al. in [17]. Hence, the security proof of the underlying signature scheme follows their proof. The proof is given by the following games between the adversary A against the signature and Challenger C. Each game is a slight modification of the preceding game in a way that the difference between the two games can be evaluated. Thus, the quality of the reduction can be easily quantified. The probability that A wins the $Game_i$ is denoted by $Pr[Success_i]$.

$\#Game_1$: C chooses a DAG G and picks a random homomorphic hash function H and a homomorphic pseudorandom generator PG and associates them with the compression vertices and expansion vertices of G, respectively. C then runs the key generation algorithm and provides the public key to A . Because C knows the entire secret key, it can easily provide any one-time signature on A's requests. The success probability of this game is ϵ, so

$$Pr[Success_1] = \epsilon \tag{1}$$

$Game_2$: As $Game_1$, but with the modification that C picks an internal node at random and keeps it as the target node. C aborts if the adversary's signature query includes an arc predeceasing the target node or if the adversary's forgery does not invert the function on this node. Since the actions of C are independent of the chosen target, the forger behaves as in $Game_1$. The success probability is at least $\frac{Pr[Success_1]}{|v|}$. Hence, the success probability in $Game_2$ is

$$Pr[Success_2] \leq \frac{\epsilon}{n} \tag{2}$$

$Game_3$: As in $Game_2$, but with the following key generation change. C selects a target node on which to invert the function and replaces the selected node's output value with y. C additionally assigns random values to all arcs leading out of the previous graph of the target node. C recalculates the public key and hands it on to A. Except for those within the predecessor graph of the target, C can generate a value for all arcs. The arcs directly going out of the predecessor nodes in a direct line of the target node are the only arcs for which C has no value. C responds to the signature question if the adversary's inquiry does not contain any arcs within the preceding graph. C aborts the signature query if this is the case. When A wins $Game_3$, the adversary has assigned a value y' and its preimage to the targeted node. C discovered a preimage to the target if $y' = y$. If $y' \neq y$, on the other hand, a collision could occur somewhere along the way to the root. As a result, when A forges, $Game_3$ success results in either a collision on H or a preimage (for some y_H or y_{PG})

$$Pr[Success_3] \leq Pr[Invert\ or\ Collision] \leq \epsilon_{PG} + \epsilon_H + \epsilon_C \tag{3}$$

It is also evident that, when A distinguishes between $Game_2$ and $Game_3$, it is distinguishing between a right evaluation of the graph consisting of the predecessors in a direct line of the goal, along with their offspring, and an equal number of uniformly chosen uniform random values. As a result, this chance is negligible if all of the predecessors are undetectable. When an efficient signature forger exists, then

$$|Pr[Success_3] - Pr[Success_2]| \leq \alpha Pr[Detect] \leq \alpha\epsilon_D \tag{4}$$

From the sequence of the games and by considering the probability of finding collision on chameleon hash ϵ_G, we have

$$\epsilon \leq (\alpha\epsilon_D + \epsilon_C + \epsilon_{PG} + \epsilon_H)n + \epsilon_G \tag{5}$$

From Equation (5), if PG and H are cryptographically secure, then the HBCS scheme is unforgeable under a chosen message attack. □

Theorem 5. *The proposed chameleon signature scheme satisfies the property of non-transferability.*

Proof. The semantic security of the proposed hash scheme in Theorem 2 implies the non-transferability of the resulting chameleon signature scheme. □

Theorem 6. *The proposed chameleon signature scheme satisfies the property of non-repudiation.*

Proof. In the case of disputes, if a verifier provides a chameleon signature forgery, then the signer can repudiate it by producing collision pairs for the judge. By Theorem 1, the chameleon hash function is collision-resistant. Hence, in case of disputes, when the signer is not able to produce valid collision pairs on the chameleon hash to the judge, he is indeed the generator of the signature. □

Theorem 7. *The proposed chameleon signature scheme satisfies the property of deniability.*

Proof. It is ensured by the Denial protocol. □

7. Discussion

Performance Analysis and Comparison: The significant parameters for a graph-based signature scheme on a DAG G are the number of internal vertices, which reveals the number of function evaluations required to compute the public key from the secret key,

the maximum number of incompatible signatures, which gives the upper bound on the size of the message space, and the maximal size of the signature.

The proposed scheme requires hash evaluations and pseudorandom evaluations for the computation of public key, signature generation, and the verification process. For a DAG with n internal vertices, the worst-case time complexity of hash evaluations and pseudorandom evaluations are $O(\frac{n}{2})$ and $O(\frac{nk}{2})$, respectively. When n_1 and n_2 are the number of vertices in G_S and G_V, respectively, the time complexity of signature generation is $O(n_1k) + O(n_2k)$. The verification process also requires the same time.

For a given graph G with n internal vertices, let $\nu(n)$ be the width of the associated poset. Table 1 shows that, for a given G of n internal vertices, there are $\nu(n)$ compatible minimal allowable sets. By optimizing n and the structure of the graph, $\nu(n)$ can be increased and hence the scheme can achieve the best possible efficiency in terms of signature size and computational costs.

Table 1. Table of n and $\nu(n)$.

n	$\nu(n)$	n	$\nu(n)$
4	2	18	66
6	5	20	86
8	8	22	117
10	14	24	147
12	20	26	190
14	33	28	232
16	45	30	289

The size of the chameleon signature σ depends on the length of $r^{'}$ and the length of λ. The length of λ depends on n_1—the number of internal vertices of G_S and the length of $r^{'}$ depends on the length of the output of the hash function H which is of k bits. The minimal allowable sets in the maximal antichain of the proposed graph construction in Algorithm 1 contain either $\frac{n_1}{2}$ or $\frac{n_1}{2} + 1$ edges. Hence, λ consists of values of either $\frac{n_1}{2}$ edges or $\frac{n_1}{2} + 1$ edges, where each value is of k-bits. Therefore, the maximum size of the signature $\sigma = (\frac{n_1}{2} + 1)k + k = (\frac{n_1}{2})k + 2k$.

Few chameleon signature schemes are constructed by combining different chameleon hash functions proposed in [2,4,18] with the standard signature algorithms (RSA, DSS, etc.) of the same public key settings, and the comparison of those schemes with the proposed work for the security level of 80 bits is shown in Table 2—for instance, the hash function defined in [19] and the pseudorandom generator defined in the [19] proposed scheme. The hash function $H : \{0,1\}^{2k} \rightarrow \{0,1\}^k$ is defined by $H(x) = Ax \bmod q$, where $A \in Z_q^{n \times 2k}$ and pseudorandom generator $PG : \{0,1\}^k \rightarrow \{0,1\}^{2k}$ is defined by $PG(x) = (g_B(s,x), g_B(s,x))$, where $g_B(s,x) = Bs + x \bmod q$ for $B \in Z_q^{n \times k}$ and a fixed secret $s \in Z_q^k$. In order to get 80-bit security, one needs to set the parameters of the functions k, n and q as $k = n \log q$ and $q = n^2$ as suggested by Micciancio Daniele in [20]. To achieve comparable security, we set the discrete logarithm based schemes $p = 1024$ and $q = 160$ bits, and the RSA scheme's RSA modulus to 1024 bits. The suggested quantum resistant system's signature size is also compared to that of a quantum resistant lattice-based chameleon signature technique [13]. The parameters q = 100,000,007 and n equals the security level λ are used to determine the signature size, as indicated in [13]. The comparison results in Table 2 show that the signature length of the proposed scheme is larger than that of non-quantum schemes. This cost can be considered as the price for the proposed scheme to be quantum safe. In addition, the proposed scheme is key-exposure free and its signature length is smaller when compared to the existing quantum resistant chameleon signature scheme proposed in [13].

Table 2. Comparison of the proposed HBCS scheme with existing chameleon signature schemes.

Scheme	System	Hard Problem	Signature Size (Bits)	KE	QS
Krawczyk et al. [2]	Num.-based	DLP	480	Yes	No
Ateniese & de Medeiros [18] + RSA	Num.-based	IFP	2048	Yes	No
Ateniese & de Medeiros [4] + DSS	Num.-based	DLP	640	No	No
Xie Dong et al. [13]	Lattice-based	SIS	5×10^9	No	Yes
Proposed HBCS	Hash-based	PR	1×10^4	No	Yes

Num: Number Theory, KE: Key Exposure, QS: Quantum Safe.

8. Conclusions

First, a new quantum-resistant hash-based chameleon hash technique with graph configuration is proposed in this study. It meets all of the desired security requirements, including collision resistance, semantic security, and key-exposure freeness. Second, utilizing the suggested chameleon hash and a graph-based one-time signature method, a privacy-preserving quantum-resistant chameleon signature technique is created. To the best of our knowledge, it is the first scheme under hash-based cryptography that could satisfy the quantum world's requirement of privacy-preserving signatures. Furthermore, the comparison of the proposed scheme with the existing chameleon signature schemes shows that it has a smaller signature length than the existing quantum-safe scheme.

Author Contributions: Conceptualization, P.T.; Formal analysis, P.T. and R.A.; Funding acquisition, G.P.J. and E.Y.; Investigation, N.A., W.C. and E.Y.; Methodology, R.A.; Project administration, G.P.J.; Resources, E.Y.; Software, N.A.; Supervision, W.C. and G.P.J.; Validation, R.A., W.C. and E.Y.; Visualization, N.A.; Writing—original draft, P.T.; Writing—review & editing, G.P.J. and E.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: N. Anbazhagan would like to thank RUSA Phase 2.0 (F 24-51/2014-U), DST-FIST (SR/FIST/MS-I/2018/17), DST-PURSE 2nd Phase programme (SR/PURSE Phase 2/38) and UGC-SAP (DRS-I)(F.510/8/DRS-I/2016(SAP-I)), Govt. of India.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chaum, D.; Van Antwerpen, H. Undeniable signatures. In *Advances in Cryptology-Crypto 89 Proceedings*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 212–216.
2. Krawczyk, H.M.; Rabin, T.D. Chameleon Hashing and Signatures. U.S. Patent 6,108,783, 22 August 2000.
3. Zhang, F.; Safavi-Naini, R.; Susilo, W. ID-Based Chameleon Hashes from Bilinear Pairings. *IACR Cryptol. ePrint Arch.* **2003**, *2003*, 208.
4. Ateniese, G.; de Medeiros, B. On the key exposure problem in chameleon hashes. In *International Conference on Security in Communication Networks*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 165–179.
5. Chen, X.; Zhang, F.; Kim, K. Chameleon hashing without key exposure. In *Information Security*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 87–98.
6. Gao, W.; Wang, X.L.; Xie, D.Q. Chameleon hashes without key exposure based on factoring. *J. Comput. Sci. Technol.* **2007**, *22*, 109–113. [CrossRef]
7. Chen, X.; Zhang, F.; Tian, H.; Wei, B.; Kim, K. Discrete logarithm based chameleon hashing and signatures without key exposure. *Comput. Electr. Eng.* **2011**, *37*, 614–623. [CrossRef]

8. Pan, P.; Wang, L.; Yang, Y.; Gan, Y.; Wang, L.; Xu, C. Chameleon hash functions and one-time signature schemes from inner automorphism groups. *Fundam. Inform.* **2013**, *126*, 103–119. [CrossRef]
9. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
10. Chang, W.L.; Vasilakos, A.V. *Fundamentals of Quantum Programming in IBM's Quantum Computers*; Springer: Berlin/Heidelberg, Germany, 2021.
11. Chang, W.L.; Chen, J.C.; Chung, W.Y.; Hsiao, C.Y.; Wong, R.; Vasilakos, A.V. Quantum Speedup and Mathematical Solutions from Implementing Bio-molecular Solutions for the Independent Set Problem on IBM's Quantum Computers. *IEEE Trans. NanoBiosci.* **2021**, *20*, 354–376. [CrossRef] [PubMed]
12. Ajtai, M. Generating hard instances of lattice problems. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 99–108.
13. Xie, D.; Peng, H.; Li, L.; Yang, Y. Homomorphic signatures from chameleon hash functions. *Inf. Technol. Control* **2017**, *46*, 274–286. [CrossRef]
14. Grover, L. A fast quantum mechanical algorithm database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219.
15. Chen, L.; Han, L.; Jing, J.; Hu, D. A post-quantum provable data possession protocol in cloud. *Secur. Commun. Netw.* **2013**, *6*, 658–667. [CrossRef]
16. Hevia, A.; Micciancio, D. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 379–396.
17. Dods, C.; Smart, N.P.; Stam, M. Hash based digital signature schemes. In *IMA International Conference on Cryptography and Coding*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 96–115.
18. Ateniese, G.; de Medeiros, B. Identity-based chameleon hash and applications. In *Financial Cryptography*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 164–180.
19. Micciancio, D. Cryptographic functions from worst-case complexity assumptions. In *The LLL Algorithm*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 427–452.
20. Micciancio, D.; Mol, P. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 465–484.