Abstract

We prove the existence of a class of functions we call \( \mathbb{T} \) (Greek Tau), each of whose members satisfies the conditions of one-way functions. Each member of \( \mathbb{T} \) is a function computable in polynomial time, with negligible probability of finding its inverse by any polynomial probabilistic algorithm. This is accomplished by constructing each \( \tau \in \mathbb{T} \) with a collection of independent universal hash functions that produce a starting coordinate and a path within a sequence of unique random bit matrices. The existence of one-way functions implies that \( P \neq NP \).

1. Introduction

This paper presents the proof of the existence of a class of functions we call \( \mathbb{T} \) (Greek Tau), each of whose members satisfies the conditions of one-way functions. Each member of \( \mathbb{T} \) is a function computable in polynomial time, with negligible probability of finding its inverse by any polynomial probabilistic algorithm. This is accomplished by constructing each \( \tau \in \mathbb{T} \) with a collection of independent universal hash functions that produce a starting coordinate and path within a sequence of unique random bit matrices.

The existence of one-way functions has been an open question in computer science. This is due to the fact that, given a candidate function, it is hard to prove that any polynomial-time random algorithm that attempts to find any member of the inverse of the candidate, has a negligible probability of success, where such probability is a function of the size of the input. To prove this, it is important to prove that no algorithm that is able to find the inverse of the candidate function runs in

---

1 This paper has been made publicly available through the World-Wide Web on April 12, 2016. The algorithms and constructions presented in this paper are part of a pending USA patent.
2 This research has been sponsored by Entevia, LLC, a limited-liability company organized under the laws of the State of Florida, USA.
polynomial time; otherwise, such algorithm can be simulated by a probabilistic polynomial-time (PPT) algorithm with full certainty. Also, the proof of the non-existence of such algorithm in P makes our proof not to rely on the assumption of the resolution of the P vs. NP problem (which would be a recursive, non-constructive proof).

Since the solution of finding the inverse of a one-way function can be verified in polynomial time (the FNP class), the existence of a one-way function means that there are problems in FNP that do not have a polynomial-time algorithm (the FP class) for finding the inverse. Thus, the existence of one-way functions implies that FP ≠ FNP, and thus P ≠ NP.

The rest of this paper is organized as follows. We start by reviewing the definition of universal hash families, a fundamental primitive of functions in T; followed by a review of one-way functions. A definition of the T class is provided, along with a proof of its complexity. A survey of related works is included. Complexity bounds for computing the inverse of τ are presented, followed by a proof that each member in T is a one-way function. A possible application in cryptography is also presented. Conclusions and references are presented at the end.

2. Definitions

A universal hash family H is defined as a set of hash functions

\[ H = \{ h : U \rightarrow M \mid U \in \{0,1\}^n, M \in \{0,1,\cdots,m\}, m < n \} \]

that satisfies the following property:

\[ \forall x,y \in U, x \neq y \Pr \left[ h(x) = h(y) \right] \leq \frac{1}{m} \]

which means that any two members of U collide with probability at most 1/m. Universal hash functions provide the guarantee that, for any S \subseteq U, and for any x,y \in S, the expected number of values of y that satisfy h(x)=h(y) is n/m [CW77].

A function f : \{0,1\}^* \rightarrow \{0,1\}^* is one-way if and only if it can be computed in polynomial time, but any probabilistic polynomial time (PPT) algorithm F will only
succeed in finding the inverse of \( f \) with \textit{negligible} probability. More formally, \( f \) is one-way iff:

\[
\Pr \left[ f(F'(f, \text{unary}(n), f(x))) = f(x) \right] < n^{-c}
\]

where:

1. \( F' \) is a PPT algorithm that attempts to find the inverse of \( f(x) \)
2. \( n = \lceil \log_2(x) \rceil \) (the number of bits in \( x \))
3. \( \text{unary}(n) \) is an all-ones bit string of size \( n \), used by \( F' \) to set an upper bound in its search
4. \( c \) is any positive integer

Intuitively, the asymptotic value of \( \Pr \) as a function of \( n \), will always be less than the asymptotic value of any polynomial of \( n \), as \( n \) tends to infinity.

\section{3. The Tau (\( \mathcal{T} \)) Functions Class}

We define the class of functions \( \mathcal{T} \) (Greek Tau) as the set of functions

\[
\mathcal{T} = \{ \tau \mid \tau: \{0,1\}^n \rightarrow \{0,1\}^n \},
\]

where each \( \tau \) is constructed as follows:

1. Construct a sequence \( M \) of \( n \) matrices \( M=(M_0, M_1, \ldots, M_n) \), each one of size \( n \times n \) where each element is a random bit, bits in each matrix are uniformly distributed, and each \( M_i \in M \) is unique.
2. Pick randomly a sequence of \( n \) hash functions \( H_{\text{row}} = (h_{\text{row}1}, h_{\text{row}2}, \ldots h_{\text{row}n}) \), from a universal hash family: \( \{ h : \{0,1\}^n \rightarrow \{0,1\}^{\log(n)} \} \)
3. Pick randomly a sequence of \( n \) hash functions \( H_{\text{col}} = (h_{\text{col}1}, h_{\text{col}2}, \ldots h_{\text{col}n}) \), from a universal hash family: \( \{ h : \{0,1\}^n \rightarrow \{0,1\}^{\log(n)} \} \)
4. Construct an \( n \times n \) matrix \( H_M \) of hash functions by picking randomly from a universal hash family: \( \{ h : \{0,1\}^n \rightarrow \{0,1\}^8 \} \)
5. Construct an \( 8 \times 2 \) matrix \( D \) with the following values, each of which represents a moving direction along coordinates in a matrix in \( M \):

\[
\begin{array}{c|cc|}
| & 1 & 0 & (\text{up}) \\
| & -1 & 0 & (\text{down})
\end{array}
\]
All hash functions constructed for $\tau$ follow Carter and Wegman’s construction [CW77] of the form:

$$((ax + b) \mod p) \mod t$$

where:

1. $p$ is an n-bit prime number chosen at random, with the condition $p > 2$
2. $a$ is an integer chosen at random, with the condition $0 < a < p$
3. $b$ is an integer chosen at random, with the condition $0 < b < p$
4. $t=n$ for all the members of $H_{\tau_{row}}$ and $H_{\tau_{col}}$; and $t=8$ for all the members of $H_M$

Each $h_i$ in any collection (sequence or matrix) $C_H$ of universal hash functions in $\tau$ is subject to the following constraints:

1. Each $h_i$ should be unique within $C_H$. That is, for any two distinct $h_i,h_j \in C_H$, $a_i \neq a_j$ $b_i \neq b_j$ and $p_i \neq p_j$.
2. Each $h_i$ should be independent from any other $h_j$ in $C_H$. That is, given two distinct $h_i,h_j$ in $C_H$, $(a_ix + b_i) \neq k(a_jx + b_j)$ for any integer $k>0$.

Once constructed, each $\tau$ maps an input $x$ in $\{0,1\}^n$ to an output $y$ in $\{0,1\}^n$ by the following algorithm $A_\tau$.

1. Let the output $y$ initially be an empty binary sequence
2. For each $i \in (1,2,\ldots,n)$:
   a. Let $hr_i$ be the $i^{th}$ hash function in $H_{\tau_{row}}$
   b. Let $hc_i$ be the $i^{th}$ hash function in $H_{\tau_{col}}$
c. Let \( r = h_r(x) + 1 \)

d. Let \( c = h_c(x) + 1 \)

e. Let \( M_i \) be the \( i \)th matrix in the bit-matrix sequence \( M \)

f. Let \((r,c)\) represent a (row, column) coordinate in \( M_i \)

g. For each \( j = (1, 2, \ldots n) \)

i. Let \( h_{d_{i,j}} \) be the hash function from the matrix \( H_{M_i} \)

ii. Let \( d = h_{d_{i,j}}(x) + 1 \)

iii. Let \( r_\Delta = D_{(d,1)} \)

iv. Let \( c_\Delta = D_{(d,2)} \)

v. Let \( r = r + r_\Delta \)

vi. Let \( c = c + c_\Delta \)

vii. If either \( r, c \) are less than one, let it be equal to \( n \)

viii. If either \( r, c \) are greater than \( n \), let it be equal to \( 1 \)

ix. Let \( b = M_i(r,c) \) (the bit at the coordinate \((r,c)\) in \( M_i \))

x. Append \( b \) to the output \( y \)

3. \( \tau(x) = y \)

**Lemma 1:** For every \( \tau \) in \( \mathcal{\Gamma} \), the algorithm \( A_\tau \) computes \( \tau \) in polynomial time.

**Proof:**

1. All the variable assignment and hashing steps are \( O(1) \)

2. Step 2 computes \( y \) in \( O(n^2) \), since:

   a. The outer loop in Step 2 is linear in \( n \), thus it is \( O(n) \).

   b. Step 2.g is linear in \( n \), thus it is \( O(n) \)

   c. The complexity of the outer loop in terms of the inner loop is \( O(n \times (n + k)) \) for some \( k > 1 \), so it is \( O(n^2) \).

3. The sum of the complexities of the sequential steps is \( O(n^2) \)

4. **Related Works**

   The construction of one-way functions has been the subject of extensive research in the past three decades, especially with applications in cryptography [L03]. Most of the possible-one-way function constructions for cryptographic applications are based
on the assumption of the existence of one-way functions. Good surveys of application of one-way hash functions in cryptography can be found in [IZM90] and [NY89].

Our work is closely related to the use of random predicates [G00] and expander graphs [CLG09] in the use of a random walk along a graph to generate the value of the output of a hash function. One obvious difference is that our proposed functions are not hash functions. Disregarding this fact, our work is related to the use of random predicates in the sense that the output of the function is a composition of bits generated from random predicates; however, our constructions are based on random walks using the output of several unique hash functions, which avoids the need to use a lookup table to store the mappings of random predicates. In the case of expander graphs, the random walk is made along a set of vertices, each of which represents an elliptic curve, and the “current” input bit is used to decide between two isogenous curves. The output of the hash is a function of the j-invariant of the last vertex of the random walk. The one-wayness of functions based on this kind of expander graphs depends on the assumption of the hardness of finding isogeneous elliptic curves, which is an NP-complete problem; thus, the proof of one-wayness is dependent on the resolution of the P vs. NP problem.

Our work is also related to previous work based on composition of universal hash functions. In [SV00] such compositions are intended to build one-way cryptographic hash functions that break a message into independent sequences for which a set of randomly-picked universal hash functions are applied and whose output concatenated to produce the hashed output. In [BP97], the composition is made by breaking the input into fixed-sized strings to which distinct hash functions are applied, each of whose output is used to compose the function’s output by application of other hashes to the outputs in cascade (either linearly or from a tree), being the final output the result of the hash function in the last stage. Notice that in neither of these works, traversals through a sequence of distinct random bit matrices are used.

5. Complexity Bounds for Computing the Inverse of \( \tau \)

Since every problem in P is also in PP (as we can always convert a deterministic Turing Machine (TM) into a Probabilistic TM where two random transitions from the
same state bring the same output and have the same next state in common), we want to prove that solving the inverse of $\tau$ is not in FP, thus justifying the need of a PPT algorithm to compute the inverse in average polynomial time.

**Lemma 2:** Any deterministic algorithm $A_{\tau}$ that attempts to find a member of the inverse of a function $\tau \in T$, has a worst-time execution time bounded by $\Omega(8^n)$.

**Proof:** Let $y = \tau(x)$ for some $x$ in the domain of $\tau$. Let $y_i$ be the $i^{th}$ bit of $y$. Each $y_i$ is produced by a bit from the corresponding matrix $M_i$. Since $A_{\tau}$ is deterministic, it has to find $n$ bits in $M$ such that for each bit $b_i$ in $M_i$ : (i) $b_i = y_i$; and (ii) a path resulting for computing the hash values from the functions $H_{\tau_{row}}$, $H_{\tau_{col}}$, and $H_M$ lead to $b_i$. In the worst case, such bits will be at the end of each matrix which brings a worst-case complexity bounded by $\Omega(n^3)$ (the size of each matrix times $n$). At some point during its execution, $A_{\tau}$ will need to find the common inverse (i.e. the intersection of all the preimages) of all hash functions that produced a walk in the path to each $b_i$. There are eight possible directions to a given point a walk, and $n$ bits, thus there are $8^n$ possible permutations. In the worst case, $A_{\tau}$ will have to try all permutations before finding the common inverse of all hash functions. Therefore, the worst-case execution time is bounded by $\Omega(n^3 + 8^n)$, which is $\Omega(8^n)$.

**Lemma 3:** Let $y = \tau(x)$ for some $x$. Let $T$-$INV$ be the problem of finding one member of the preimage $\tau^{-1}\{y\}$. $T$-$INV$ cannot be reduced to FSAT in polynomial time.

**Proof 1:** Let $R = \{ <y_1,y_2,\ldots,y_n,x_1,x_2,\ldots,x_n> | y = y_1y_2\ldots y_n, x = x_1x_2\ldots x_n, x \in \tau^{-1}\{y\} \}$ be the set of tuples that correspond to the inverse mappings of the image of $\tau$. Since the problem is to find just one member of the preimage $\tau(y)$, we are interested in a minimal set of inverse mappings from the preimage, i.e. for each $y = \tau(x)$, represent one mapping $x \in \tau^{-1}\{y\}$. Let $R' \subset R$ represent a minimal set of mappings from the preimage $\tau^{-1}\{y\}$. Since $\tau$ is a total function, every value $x \in \tau^{-1}\{y\}$ maps some member of the domain of $\tau$. Thus, the number of tuples in $R'$ is $\Omega(2^n)$, each of which corresponds to an inverse mapping of $\tau$. Thus, any algorithm $A_R(R')$ that reduces $R'$ to FSAT will have to process $\Omega(2^n)$ tuples, which will take $\Omega(2^n)$ time.

**Proof 2:** Let $C_\tau_i$ be a circuit that computes the $i^{th}$ output bit $y_i$ of $\tau(x)$. $C_\tau_i$ needs to encode each possible path $S=(r_i,c_i,d_{i1},d_{i2},\ldots,d_{in})$ that will output either $y_i=1$ or $y_i=0$. 


Since the bits in the matrix $M_i$ are uniformly distributed, there need to be $n^2/2$ of such encodings for either value of $y_i$. For each $y_i$, there are $8^n$ possible paths, thus there are $\Omega(n^2 8^n)$ possible sub-circuits that encode each path. This implies that any SAT formula will need at least $n^2 8^n$ clauses. Thus, the size of the formula will be exponential in $n$. Again, any algorithm $A_{R1}(C_1)$ that reduces $C_1$ to FSAT will have to process $\Omega(2^n)$ clauses, which will take $\Omega(2^n)$ time.

**Corollary 1:** T-INV is not in FP.

### 6. All members of $\tau$ are One-Way Functions

In this section, we shall prove that all members of $T$ are one-way functions.

**Lemma 4:** Let $y = \tau(x)$ for some $x \in \{0, 1\}^n$. Let $H_M$ be the matrix of universal functions used to construct $\tau$, and let $H_{Mi}$ be the $i$th row of the $H_M$. Let $h_i \in H_{Mi}$, where $i \in \{1, 2, \ldots, n\}$, let $y_i$ be the $i$th bit of $y$. Let $(d_{i1}, d_{i2}, \ldots, d_{in})$ be the output of each respective $h_i$ for an $x$ in the domain of $\tau$, which defined the path to $y_i$. Let $F_i$ be the event of finding at random one element from the preimage $\tau^-(\{y_i\})$ with a path to $y_i$. The probability of $F_i$ is given by:

$$\Pr[F_i] = 1/8^n$$

**Proof:** Since $h_i$ in $H_M$ is member of a universal family of hash functions, it is guaranteed that the size of the preimage $|h_i^-(\{y_i\})| = 2^n/8$. The size of the domain of $h_i$ is $2^n$, thus the probability of finding an $x$ in the preimage $h_i^-(\{y_i\})$ is $2^n/(2^n/8) = 1/8$. Since all hash functions are independent, the probability of finding an $x$ common to all $h_i$ is given by:

$$\Pr[F_i] = \prod_i \Pr(x \in h_i^-(\{y_i\})) = \prod_i (1/8) = 1/8^n$$

$i = \{1, 2, \ldots, n\}$

**Lemma 5:** Let $y = \tau(x)$, $y_i$, $H_M$, $H_{Mi}$ and $h_i$ be as defined in Lemma 4. Let $F_i$ be the event of finding at random one element from the preimage $\tau^-(\{y\})$ with a path to $y_i$. Let $F_j$ be the event of finding at random one element from the preimage $\tau^-(\{y\})$ with a path to $y_j$, and $i \neq j$. The probability of $F_i$ given $F_j$, when $F_i \neq F_j$, is given by:

$$\Pr[F_i | F_j] = \alpha \cdot 1/8^n$$

for some constant $0 < \alpha < 1$.

**Proof:** We know that:
\[
\Pr[F_i \mid F_j] = \frac{\Pr[F_i \cap F_j]}{\Pr[F_j]}
\]
and \(\Pr[F_k] = 1/8^n\) for all \(k = \{1, \ldots, n\}\), and that \(F_j \not\subset F_i\). Since we are assuming some degree of dependency between \(F_i\) and \(F_j\) (otherwise the conditional probability would be zero), it follows that
\[
\Pr[F_i \cap F_j] = \alpha \cdot 1/8^n \cdot 1/8^n
\]
for some constant \(0 < \alpha < 1\). Replacing in the equation yields to:
\[
\Pr[F_i \mid F_j] = \alpha \cdot 1/8^n
\]

**Lemma 6:** Let \(y = \tau(x)\) for some \(x\) in the domain of \(\tau\). Let \(F_i\) be the event of finding at random one element from the preimage \(h_i(\{y_i\})\). Let \(F_{-i}\) be intersection:
\[
F_{-i} = \bigcap_{k \neq i} F_k
\]
The probability of \(F_{-i}\) is given by:
\[
\Pr[F_{-i}] = \beta \cdot (1/8^{(n^2-n)})
\]
for some constant \(0 < \beta < 1\).

**Proof:** From Lemma 4, we know that \(\Pr[F_k] = 1/8^n\) for all \(k = \{1, \ldots, n\}\). Since we are assuming some dependency among events, the probability of the intersection is given by:
\[
\Pr[F_{-i}] = \beta \prod_{k \neq i} \Pr[F_k]
\]
for some constant \(0 < \beta < 1\). Thus,
\[
\Pr[F_{-i}] = \beta \cdot (1/8^n)^{(n-1)}
\]
which is equivalent to
\[
\Pr[F_{-i}] = \beta \cdot (1/8^{(n^2-n)})
\]

**Lemma 7:** Let \(F_i\) and \(F_{-i}\) be as defined in Lemmas 5 and 6. The probability of \(F_i\) given \(F_{-i}\) is given by:
\[
\Pr[F_i \mid F_{-i}] = \gamma \cdot 1/8^n
\]

**Proof:** We know that
\[
\Pr[F_i \mid F_{-i}] = \Pr[F_i \cap F_{-i}] / \Pr[F_{-i}]
\]
From Lemma 5
\[
\Pr[F_i \cap F_{-i}] = \alpha \cdot 1/8^n \cdot 1/8^{(n^2-n)}
\]
for some constant \(0 < \alpha < 1\). From Lemma 6:
\[
\Pr[F_{-i}] = \beta \cdot (1/8^{(n^2-n)})
\]
Replacing in the formula above gives:

\[ \Pr[F \mid F \neq i] = \alpha \cdot 1/8^n \cdot 1/8^{(n^2-n)} / (\beta \cdot (1/8^{(n^2-n)})) \]

which brings:

\[ \Pr[F \mid F \neq i] = \gamma \cdot 1/8^n \]

where \( \gamma = \alpha / \beta \) and \( 0 < \gamma < 1 \).

**Lemma 8:** Let \( y, y_i \) and \( F_i \) be as defined in Lemma 4. Let \( F \) be the event of finding at random one member of the preimage \( \tau(\{y\}) \). The probability of \( F \) is given by:

\[ \Pr[F] = \Pr[\cap F_i] \]

\[ i \]

**Proof:** Each \( y_i \) is produced independently by a set of \( x \) in the preimage \( \tau(\{y\}) \). However, due to the universal-hash guarantee and the distinct parameters of each hash function in \( \tau \), there exists an \( x \) that produces a bit in \( y \) but does not produce same bit value in another bit of \( y \). Thus, the preimage \( \tau(\{y\}) \) is given by the intersection of the independent preimages for each bit \( y_i \) in \( y \).

With these lemmas proved, we are now in a position to prove that all members of \( T \) are one-way functions.

**Theorem 1:** Let \( y = \tau(x) \) for some \( x \in \{0,1\}^n \). Let \( F \) be as defined in Lemma 7. The probability of \( F \) is bounded by:

\[ \Pr[F] < n^{-c} \]

where \( c \) is any positive integer.

**Proof:** From Lemma 8, we know that the probability of \( F \) is given by:

\[ \Pr[F] = \Pr[\cap F_i] \]

\[ i \]

which is equivalent to:

\[ \Pr[F] = \Pr[F_1 \cap F_2 \ldots \cap F_n] \]

Given two distinct \( F_i \) and \( F_j \), the probability of their intersection is given by:

\[ \Pr[F_i \cap F_j] = \Pr[F_i \mid F_j] \Pr[F_j] \]

By the chain rule for multiple intersections of sets, the probability of \( F \) in terms of the intersection of all \( F_i \) is given by:

\[ \Pr[F] = \Pr[\cap F_i] = \Pr[F_n \mid F_{n-1} \cap F_{n-2} \ldots F_1] \cdot \Pr[F_{n-1} \mid F_{n-2} \cap F_{n-3} \ldots F_1] \ldots \cdot \Pr[F_1] \]
Since $\Pr[F_i] = 1/8^n$ for any $F_i$ (Lemma 4), then:

$$\Pr[\cap F_i] = \Pr[F_n | F_{n-1} \cap F_{n-2} \ldots F_1] \cdot \Pr[F_{n-1} | F_{n-2} \cap F_{n-3} \ldots F_1] \cdot \ldots \cdot 1/8^n$$

We proved in Lemma 7 that $\Pr[F_i | F_{-i}] = \gamma \cdot 1/8^n$ for some constant $0 < \gamma \leq 1$ and any $F_i$ that is the intersection of all $F_j \neq F_i$. Thus, $\Pr[F_i | F_{-i}] = 1/\beta_i$ for all $F_i$, for some constant $\beta_i > 1$. Replacing each term in the equation above with its equivalent $1/\beta_i$:

$$\Pr[\cap F_i] = 1/8^n \Pi(1/\beta_i) \quad \text{i = \{1,2,\ldots,n\}}$$

Since each term in the $\Pi$ operation is less than one, then:

$$\Pr[\cap F_i] \leq 1/\alpha^n \quad \text{i = \{1,2,\ldots,n\}}$$

where $\alpha = \max(\beta_1, \beta_2, \ldots, \beta_n)$. Therefore,

$$\Pr[F] \leq 1/\alpha^n$$

for some constant $\alpha > 1$. Since $\Pr[F]$ is inversely exponential in $n$, it follows that:

$$\Pr[F] < n^{-c}$$

**Corollary 2:** Each $\tau \in T$ is a one-way function.

### 7. Application in Cryptography

It is possible to construct a cryptographic one-way hash function $h(x)$ from a function in $T$. The idea is to define $h : \{0,1\}^n \rightarrow \{0,1\}^m$, $g : \{0,1\}^n \rightarrow \{0,1\}^m$ and $\tau : \{0,1\}^m \rightarrow \{0,1\}^m$, where $n > m$, and $g$ is a compression function; then $h(x) = \tau(g(x))$.

Since each $\tau$ has a unique set of parameters (universal hash constants and bit matrices), a signing entity can define its own distinguishing $\tau$, and either: (i) send the hash value along with the parameters; or (ii) send a reference to a resource that provides the parameters. A receiver can authenticate a digital signature by verifying its value after applying $\tau$ with the verified parameters.
One possible issue with τ is that it takes $O(n^2)$ time to compute, which might not perform well for large n. A possibility to improve performance is to limit the size of the bit matrices to a constant $k << n$. This won’t affect the performance of computing the inverse, as there will still be $8^k$ possible paths that lead to an output bit; as long as the value k is not too small.

8. Conclusion

We have presented the proof of the existence of a class of functions $T$, each of whose members satisfies the conditions of one-way functions. Each member in $T$ is constructed with a collection of independent universal hash functions that produce a starting coordinate and path within a sequence of unique bit matrices. A proof of the exponential lower bound of worst-case complexity of finding the inverse of members of $T$ was presented. We also proved that the problem of finding the preimage of each member in $T$ is not polynomial-time reducible to FSAT, thus precluding any FNP-completeness. We also proved that any random algorithm that attempts to find the inverse of any function in $T$ has negligible probability of success, thus proving that all members of $T$ are one-way functions. A possible application is cryptography was also presented.

9. References

[BP97] Bellare, Mihir, and Phillip Rogaway. "Collision-resistant hashing: Towards making UOWHF's practical." Advances in Cryptology—CRYPTO'97. Springer Berlin Heidelberg, 1997. 470-484.

[CLG09] Charles, D.X., Lauter, K.E. and Goren, E.Z., “Cryptographic Hash Functions from Expander Graphs”, Journal of Cryptology, 2009, Volume 22, Number 1, Page 93

[CW77] Carter, Larry; Wegman, Mark N. (1979). "Universal Classes of Hash Functions". Journal of Computer and System Sciences 18 (2): 143–154. doi:10.1016/0022-0000(79)90044-8. Conference version in STOC'77.

[G00] Goldreich, O.; Candidate One-Way Functions Based on Expander Graphs, 2000
[IZM90] Imai, Yuliang Zheng Tsutomu Matsumoto Hideki. "Provably Secure One-Way Hash Functions." [link](https://cis.uab.edu/yzheng/publications/files/wcis90-1wayhash.pdf)

[L03] Levin, Leonid A. "The tale of one-way functions." Problems of Information Transmission 39.1 (2003): 92-103.

[NY89] Naor, Moni, and Moti Yung. "Universal one-way hash functions and their cryptographic applications." Proceedings of the twenty-first annual ACM symposium on Theory of computing. ACM, 1989.

[SV00] Shoup, Victor. "A composition theorem for universal one-way hash functions." Advances in Cryptology—EUROCRYPT 2000. Springer Berlin Heidelberg, 2000.