The M3dcrypt Password Hashing Function

Isaiah Makwakwa
imakwakwa@gmail.com

Abstract

M3dcrypt is a password hashing function built around the Advanced Encryption Standard (AES) algorithm and the arcfour pseudorandom function. It uses up to 256-bit pseudorandom salt values and supports 48-byte passwords.

1 Introduction

1.1 Properties of the Password Space

A password is an authentication token generated by some rule(s) as a sentential form over a finite alphabet. Therefore, given an access control system A, the password space $P_A \subseteq \Sigma^*$ for A is the set

$$P_A = \{x \in \Sigma^* : R_A(x) = ACCEPT\},$$

where $R_A : \Sigma^* \to \{ACCEPT, REJECT\}$ validates conformity to a finite set of password rules [based on the prevailing security policy] and Σ is the password alphabet.

Since Σ^* contains infinite length strings, one expects that in practice,

$$P_A \subseteq \bigcup_{MIN_p \leq i \leq MAX_p} \Sigma^i \subsetneq \Sigma^*$$

for some $MAX_p \in \mathbb{N}$ less than or equal to the maximum string length on A and

$$MIN_p = min\{|x| : x \in \Sigma^*, R_A(x) = ACCEPT\}.$$

Therefore, P_A is clearly finite. Alternatively, for any $i \geq 1$, let $trunc_i : \Sigma^* \to \bigcup_{1 \leq j \leq i} \Sigma^j$ be the string truncating function defined by

$$trunc_i(x) = \begin{cases} x' = x_0x_1x_2\cdots x_{i-1} & \text{if } |x| > i, \\ x & \text{otherwise,} \end{cases}$$

where $x_i \in \Sigma$, for all $0 \leq i \leq |x| - 1$. Let $\simeq_i$ denote the equivalence relation on Σ^* defined by

$$u \simeq_i v \text{ iff } trunc_i(u) = trunc_i(v)$$

for any $u, v \in \Sigma^*$.

Define

$$P_A^{'} = \left\{ [x] \in \Sigma^* / \simeq_{MAX_p} : x \in \bigcup_{MIN_p \leq i \leq MAX_p} \Sigma^i,\ R_A(x) = ACCEPT \right\},$$

where $\Sigma^* / \simeq_{MAX_p}$ is the set of all equivalence classes in Σ^* under $\simeq_{MAX_p}$. Then, clearly, there is a bijection between P_A and $P_A^{'}$. Most importantly, in relation to R_A [and thus the security policy], all $y \in [x]$ provide equivalent security. Thus, we could define P_A over infinite length strings as follows

$$P_A = \left\{ x \in \Sigma^* : R_A(trunc_{MAX_p}(x)) = ACCEPT \right\}.$$

Since the password space membership problem is *decidable* in polynomial time (otherwise, *proactive password checking* may not admit any password in the system's lifetime and inadmissible passwords may be indefinately acceptable in *passive password checking* [24, 23]), an *efficient* algorithm for R_A must exist. That is, there exists a Turing Machine M_A (implementing R_A) and a polynomial Q_A such that for all $x \in \Sigma^*$

(i) $x \in P_A$ iff $M_A(x)$ accepts.

(ii) M_A halts after at most $Q_A(|x|)$ steps [9].

This implies that P_A is at most a *type-2 language* [10].

Example 1.1. *Let P_A be the language rejected by the k^{th}-order Markov model [m,A,T,k] in the Davies-Ganesan proactive checker [7] where m is the number of states in the Markov model, A is the state space, T is the matrix of transition probabilities and $k \geq 1$ is the order of the model. Then P_A is a context-free (type-2) language.*

Proof. We prove by constructing a Pushdown Automata (PDA) M_A which accepts P_A by empty stack as follows.

We assume that the password alphabet $\Sigma = \{a_0, a_1, a_2, \cdots, a_n\}$ is finite or the Markov model is unworkable (for example, no polynomial time algorithm can compute T [7] and no group of earthlings can realistically remember all the available symbols).

For the stack alphabet we consider the following. Let

$$\Gamma^{'} = \bigcup_{2 \leq i \leq k+1} \Sigma^i$$

and $\Gamma^{''}$ be the set obtained by relabeling the elements of $\Gamma^{'}$ using some lexicographical ordering on Σ i.e.

$$\Gamma^{''} = \left\{ \alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{|\Gamma^{'}|-1} \right\},$$

where $\alpha_i < \alpha_{i+1} \in \Gamma^{'}$ for all $0 \leq i \leq \Gamma^{'} - 2$. Then the stack alphabet, Γ, is the set

$$\Gamma = \left\{ A_0, A_1, A_2, \cdots, A_{|\Gamma''|-1} \right\} \cup \Sigma \cup \{Z_0\},$$

where Z_0 is the initial stack symbol. Therefore, Γ is finite (since Σ is finite). Further, let $\varphi : \Gamma \to \Gamma^{''} \cup \Sigma \cup \{Z_0\}$ denote the symbol conversion function defined by

$$\varphi(x) = \begin{cases} \alpha_i & \text{if } x = A_i \in \Gamma \setminus (\Sigma \cup \{Z_0\}), \\ x & \text{otherwise.} \end{cases}$$

Let $T^{'} : \Gamma \to [0, 1] \cup \{-1\}$ denote the map defined by

$$T'(x) = \begin{cases} T(\varphi(x)) & \text{if } \varphi(x) \in \Sigma^{k+1}, \\ -1 & \text{otherwise,} \end{cases}$$

and define

$$Q_T = \left\{ q_{T'(x)} : x \in \Gamma \right\}.$$

Let

$$Q \subseteq \bigcup_{0 \leq i \leq \infty} q_{-1} Q_T^i$$

denote the set of all possible states in M_A. Consider the map $\rho : Q_T \to [0, 1] \cup \{-1\}$ defined by $\rho(q_y) = y$ for any $q_y \in Q_T$.

Clearly, ρ allows recovery of all the probabilities associated with each state in Q (in relation to the transition probabilities of the n-grams in Γ). For example, ρ induces the natural extension $\hat{\rho} : Q \to [-1, 1]$ defined by

$$\hat{\rho}(p) = \rho(p_0)\rho(p_1)\rho(p_2) \cdots \rho(p_n)$$

for any $p = p_0 p_1 p_2 \cdots p_n \in Q$, $p_i \in Q_T$ for all i. Thus, states associated with elements of $\Gamma \setminus \Gamma^{k+1}$, q_{-1}, only contribute the sign.

Further, since we intend to accept by empty stack, we can set $F = \emptyset$. For state transitions, $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_{fin}(Q \times \Gamma^*)$, where $\mathcal{P}_{fin}(B)$ denotes the set of all finite subsets of B, see **Table 1** below.

Clearly, every element $x \in \Sigma^*$ such that $|x| > k$ can be broken down into at most $|x| - (k+1) + 1 = |x| - k$ $k+1$-grams [7] and any such break down leaves a tail of exactly k alphabet symbols. An optimal (thus, efficient) machine can therefore parse x using $|x| - k$ $k+1$-grams and k n-grams ($n \in \{k - i : 0 \leq i \leq k - 1\}$) of decreasing length for the tail resulting in a total of $|x| - k + k = |x|$ steps. By codifying each processing step as a state in M_A using a concatenation of transition probabilities of n-grams ($n \in \{k + 1 - i : 0 \leq i \leq k - 1\}$) so far processed, we note that after exactly $|x|$ states the Turing machine (or PDA in our case) M_A arrives in a state where it either accepts x or has an undefined transition or enters into a non-terminating loop. Therefore, we expect the maximal length of any representation of a state in M_A to be exactly MAX_p (including the initial state q_{-1}).

Transition	Possible Moves	Conditions
$\delta(q_{-1}, \epsilon, Z_0)$	$\{(q_{-1}, A_k Z_0)\}$	$A_k \in \Gamma \setminus (\Sigma \cup \{\epsilon\} \cup \{Z_0\})$.
$\delta(q, a_i, A_j)$	$\{(qq_{T'(A_j)}, a_i A_k)\}$	$k \geq 2$, $T^{'}(A_j) \neq -1$, $\varphi(A_j) = a_i a_1^{'} a_2^{'} \cdots a_k^{'}$, $\varphi(A_k) = a_1^{'} a_2^{'} \cdots a_k^{'} a_l$, $a_l \in (\Sigma \cup \{\epsilon\})$, $a_i, a_m^{'} \in \Sigma, 1 \leq m \leq k$.
$\delta(q, a_i, A_j)$	$\{(qq_{T'(A_j)}, a_i A_k)\}$	$k = 1$, $T^{'}(A_j) \neq -1$, $\varphi(A_j) = a_i a_1^{'}$, $\varphi(A_k) = a_1^{'} a_l$, $a_i, a_1^{'}, a_l \in \Sigma$.
$\delta(q, a_i, A_j)$	$\{(qq_{T'(A_j)}, a_i a_l)\}$	$k = 1$, $T^{'}(A_j) \neq -1$, $\varphi(A_j) = a_i a_l$, $a_i, a_l \in \Sigma$.
$\delta(q, a_i, A_j)$	$\{(qq_{T'(A_j)}, a_i A_k)\}$	$T^{'}(A_j) = -1, \varphi(A_j) \notin \Sigma^2$, $\varphi(A_j) = a_i a_1^{'} a_2^{'} \cdots a_{\lvert\varphi(A_j)\rvert-1}^{'}$, $\varphi(A_k) = a_1^{'} a_2^{'} \cdots a_{\lvert\varphi(A_j)\rvert-1}^{'}$, $a_i, a_m^{'} \in \Sigma, 1 \leq m \leq \lvert\varphi(A_j)\rvert - 1$.
$\delta(q, a_i, A_j)$	$\{(qq_{T'(A_j)}, a_i a_l)\}$	$T^{'}(A_j) = -1, \varphi(A_j) \in \Sigma^2$, $\varphi(A_j) = a_i a_l$, $a_i, a_l \in \Sigma$.
$\delta(q, a_i, a_i)$	$\{(q, \epsilon)\}$	$a_i \in \Sigma$
$\delta(q, \epsilon, Z_0)$	$\{(q, \epsilon)\}$	$\frac{\frac{ln(-\hat{\rho}(q))}{\lvert q\rvert-1} - \mu}{\sigma} \leq -2.6$.
$\delta(q, \epsilon, Z_0)$	$\{(q, Z_0)\}$	$\frac{\frac{ln(-\hat{\rho}(q))}{\lvert q\rvert-1} - \mu}{\sigma} > -2.6$.

Table 1: *Table for M_A's Transition Function δ.*

We can thus write

$$Q = \bigcup_{0 \leq i \leq MAX_p - 1} q_{-1} Q_T^i.$$

Therefore, Q is finite and the PDA $M_A = (Q, \Sigma, \Gamma, \delta, q_{-1}, Z_0, F)$ accepts any $x \in P_A$ by empty stack in at most $Q_A(|x|)$ steps for some polynomial Q_A as required and $P_A = N(M_A)$ [8, 10]. □

1.2 User Induced Distribution on P_A

It is known that the user induced probability distribution D on P_A has low entropy. Therefore, every *password hashing function* F over P_A admits an attacker A_D called the *dictionary attacker*, whose main tool is a set $U_{P_A} \subsetneq P_A$ such that

$$Pr[p \in U_{P_A} \mid p \xrightarrow{D} P_A] \geq 1 - \epsilon$$

for some fraction $0 \leq \epsilon < 1$.

Secondly, D potentially increases F's vulnerability to brute force attacks since the effective password space is strictly lower than $|P_A|$ i.e. $H(D) < log_2|P_A|$ where H is the Shannon entropy function [21, 17, 7, 26].

Let P_A be the password space for an access control system A, and $\chi_S : \mathcal{P}_{fin}(P_A) \to [0, 1]$ denote the probability function,

$$\chi_S(U_{P_A}) = \sum_{p \in U_{P_A}} D(p),$$

that assigns probabilities to finite subsets of P_A according to the probability distribution D. Then, for $t \geq 1$, let $\chi_D(t)$ be defined by

$$\chi_D(t) = \max \left\{ \chi_S(U_{P_A}) : U_{P_A} \in \mathcal{P}_{fin}(P_A),\ |U_{P_A}| \leq t \right\}.$$

Therefore, $\chi_D(t)$ is the success probability for an optimal dictionary attack using a dictionary of size at most t. Thus, any password hashing function for the distribution D, is at most a $(t, \chi_D(t))$-secure function. This represents ideal security against password guessing attacks [26].

However, in practice, this creates insurmountable problems for the designer of password hashing functions since it is impossible to obtain *apriori* information about the probability distribution D and thus about $\chi_D(t)$ for any $t \geq 1$. Fortunately, the following result - Theorem 4 of [26] - shows that one only needs prove security for the uniform distribution.

Theorem 1.2. *Let f be a password hashing function that is (t, ϵ)-secure for uniformly distributed inputs. Then, for every distribution D on P_A, f is a $(t, \chi_D(\epsilon|P_A|))$-secure password hashing function for D.*

Therefore, since $\chi_D(t)$ is a monotonically increasing function of t, we have that

$$\chi_D(\epsilon|P_A|) \leq \frac{\epsilon|P_A|}{t} \chi_D(t).$$

Thus, any password hashing function which is (t, ϵ)-secure for the uniform distribution does not have ideal security against password guessing attacks by a factor of at most $\frac{\epsilon|P_A|}{t}$. In short, secure password hashing functions are those with ϵ as close to $\frac{t}{|P_A|}$ as possible.

Therefore, secure password schemes involve either of the following.

[RQ1a] randomly generated passwords of sufficient length (depending on some threat model) [21].

[RQ1b] some combination of

- (i) user and/or computer generated passwords
- (ii) rules and routines to ensure generated passwords meet some minimum entropy threshold (depending on the threat model) [24, 7, 20]

(iii) some elements of key stretching to increase the time complexity for both exhaustive and dictionary attacks [13].

Requirement [RQ1a] represents the fact that under the uniform distribution, $\chi_D(t) = \frac{t}{|P_A|}$. Thus, $\chi_D(\epsilon|P_A|) = \epsilon$ i.e. the dictionary attacker does no better than a randomised inversion algorithm.

On the other hand, [RQ1b](ii) ensures that, to the extent possible, user and/or computer generated passwords model some distribution with higher entropy and thus that $\chi_D(t)$ is small for all reasonably sized password dictionaries $t \geq 1$.

Finally, [RQ1b](iii) ensures that for fixed t the attack probability $\chi_D(t)$ is quantifiably reduced thus ensuring a significant reduction in the resource envelope available to the dictionary [and/or brute force] attacker. This is particularly true if one views t as the amount of time for t complete iterations of the password hashing function.

Clearly, the above is sufficient. For example, hosts with tamper resistant password modules that act as access control oracles [i.e. authentication requests are treated as oracle queries to which the module only responds to indicate success or failure] require no transformation on stored password values - a design similar to the IBM's *Secret-Key Management Protocol* [23].

1.3 Password Hashing Functions

Presently, however, such access control systems are unavailable on all but a few systems. Therefore, in general, the following requirement on password schemes as postulated in [17, 9, 26] is necessary:

[RQ2] storage of a strong one-way transform of the user password

which requires the use of a strong one-way function $F : P_A \to Ran(F)$, called the password hashing function, where $Ran(F)$ denotes the range of F.

However, initial analysis in [17] and further analyses in [15, 24, 21] show that the user induced probability distribution D on P_A allows precomputation of tables of password dictionaries which act as inversion oracles (return either the *user password* or *failure* to each query) once password files become available. This leads to the further postulate

[RQ3] the password hashing function F must non-trivially depend on a random auxiliary input, called *salt*, of sufficient length to preclude any precomputation of password dictionaries [17, 21].

However, on the other hand, any strong one-way function $f : I \times Prim(f) \to Ran(f)$ (where $Prim(f)$ is the primary non-auxiliary input) that non-trivially depends on auxiliary input from some nonempty set I spawns a *collection of strong one-way functions* $f_c = \{f_i : Prim(f) \to Ran(f)\}_{i \in I}$, where for any element $x \in Prim(f)$, $f_i(x) = f(i, x) \in Ran(f)$.

Note that it is quite possible that f is easy to invert on a few instances f_j, $j \in E$ (for some subset $E \subsetneq I$). However, we require that those instances be such that $\frac{|E|\times|Prim(f)|}{|I|\times|Prim(f)|} = \frac{|E|}{|I|} \leq \nu_A\left(log_2|Prim(f)|\right)$ for some negligible function ν_A. Otherwise, f is not a strong one-way function. Formally, we have the following [9].

Definition 1.3. *Let I be a set of indices and for any $i \in I$, let D_i and R_i be finite. A collection of strong one-way functions is a set $f = \{f_i : D_i \to R_i\}_{i\in I}$ satisfying the following conditions.*

(1) There exists a Probabilistic Polynomial Time (PPT) algorithm S_1 which on input 1^k outputs $i \in \{0,1\}^k \cap I$.

(2) There exists a PPT algorithm S_2 which on input $i \in I$ outputs $x \in D_i$.

(3) There exists a PPT A_1 such that for $i \in I$ and $x \in D_i$, $A_1(i,x) = f_i(x)$.

(4) For every PPT algorithm A there exists a negligible function ν_A such that $\forall k$ large enough

$$Pr\left[f_i(z) = y : i \xleftarrow{\$} I; x \xleftarrow{\$} D_i; y \leftarrow f_i(x); z \leftarrow A(i,y)\right] \leq \nu_A(k)$$

where the probability is taken over choices of i and x and the coin tosses of A.

Clearly, since $\mathcal{P} \subseteq \mathcal{BPP}$ PPT algorithms S_1, S_2 and A_1 *may* require some coin tosses. Furthermore, 1^k represents the input length (i.e. k) in unitary form and, therefore, ν_A is a function of input rather than output length.

In essence, Definition 1.3 claims that [a collection of] strong one-way functions exist if $\mathcal{BPP} \neq \mathcal{NP}$, specifically that there are languages in $\mathcal{NP}$ not in $\mathcal{BPP}$ [9].

However, the above definition requires some qualification. In particular, certain security models allow the possibility of stronger adversaries capable of making multiple queries under some parameter modifying function e.g. the related-key attack in symmetric key ciphers [23, 13] and the parameter modifying attacks of [27]. Therefore, a collection of strong one-way functions suitable for such models require proof of security against parameter modifying adversaries (i.e. *related-I attacks*). This is analogous to the notion of *Strongly Secure Key Derivation Functions* postulated in [27].

Claim 1.1, for example, shows that there exists a set of strong one-way functions $f_c^* = \{f_i : Dom(f_i) \to Ran(f_i)\}_{i\in I}$ (where $Dom(f_i)$ is the domain of f_i) such that f_i non-trivially depends on $i \in I$, I a nonempty set, which does not consist a collection of strong one-way functions under related-I attacks.

Claim 1.1. *There exists a set of strong one-way functions $f_c^* = \{f_i : Dom(f_i) \to Ran(f_i)\}_{i\in I}$ such that f_i non-trivially depends on $i \in I$, I a nonempty set, which does not consist a collection of strong one-way functions under parameter modifying adversaries.*

Proof. We prove by counter example. Suppose the contrapositive holds.

Let $I = \mathbb{Z}_2^{48}$ and for each $i \in I$ (called *salt* for brevity), let g_i be the Data Encryption Standard (DES) algorithm variant (note that for simplicity we follow the standard cryptanalytic practice of ignoring the initial and final permutations IP and IP^{-1} respectively since they are of no cryptographic value) defined by

$$\begin{aligned} g_{i,k} &= \pi_{16}^i \circ \left(\bigcirc_{j=0}^{15} \pi_j\right) \\ &= \pi_{16}^i \circ \sigma \circ DES_k \end{aligned}$$

where $k \in \mathbb{Z}_2^{56}$ is the DES master key, π_j is the $(j+1)^{th}$ DES round function, DES_k is the DES encryption algorithm with master key k and σ swaps the 32-bit halves of its argument.

Further, let $\pi_{16}^i : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64}$ denote the permutation on 64-bits defined by

$$\pi_{16}^i(x_L, x_R) = \left(x_L \oplus \left(P \circ \gamma \circ \vartheta_i \circ \psi_{16} \circ E(x_R)\right), x_R\right)$$

for all $(x_L, x_R) \in \left(\mathbb{Z}_2^{32}\right)^2 \equiv \mathbb{Z}_2^{64}$, $E : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{48}$ the *DES Expansion Permutation* function, $P : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$ the *DES round Permutation* function, $\gamma : \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32}$ an array of 8 6×4 DES round S-boxes, $\vartheta_i : \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{48}$ the i^{th} *salt addition function* defined by $\vartheta_i(x) = x \oplus i$ and $\psi_{16} : \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{48}$ the 17^{th} round key addition function defined by $\psi_{16}(x) = x \oplus k_{16}$ for all $x \in \mathbb{Z}_2^{48}$.

Let the key schedule $k_j, 0 \le j \le 16$, for $g_{i,k}$ be defined by

$$\begin{aligned} k_j &= (j+1)^{th} \text{ round DES subkey for master key } k, 0 \le j \le 15 \\ k_{16} &= PC2\left(rROT_2(C_{15}) || rROT_2(D_{15})\right) \end{aligned}$$

where $PC2$ is the DES key schedule *Permuted Choice Two* function, $rROT_2$ *right circular shifts* its argument by 2 bits, $||$ is the string concatenation function, C_{15} and D_{15} are the 28-bit outputs of the 16^{th} round DES key schedule master key transformation (*round dependent left circular shifts*) [24, 23].

It is clear that for any fixed $i \in I$, the function $f_i : \mathbb{Z}_2^{56} \to \left(\mathbb{Z}_2^{64}\right)^4$ defined by

$$f_i(k) = \left(g_{i,k}(0), g_{i,k}(1), g_{i,k}(2), g_{i,k}(3)\right)$$

is a strong one-way function since g_i is essentially a DES wrapper function and ϑ_i is a linear permutation (for example see [9, 26, 23]) - that is, discounting hardware changes since the year 2000.

For clarity, we adopt some notation from the differential attacks in [3] as follows. Let

- $S\ell$: denote the ℓ^{th} DES round function S-box, $0 \le \ell \le 7$
- $S\ell_{Er}$: denote the ℓ^{th} six-bit word after the E expansion permutation of the r^{th} round, $0 \le r \le 16$, $0 \le \ell \le 7$
- $S\ell_{S16}$: denote the ℓ^{th} six-bit word of the salt value j for each function g_j, $j \in \mathbb{Z}_2^{48}$, $0 \le \ell \le 7$
- $S\ell_{Kr}$: denote the ℓ^{th} six-bit word of the r^{th} round subkey, $0 \le r \le 16$, $0 \le \ell \le 7$

$S\ell_{Or}$: denote the four-bit output of the ℓ^{th} S-box of the r^{th} round, $0 \leq r \leq 16$, $0 \leq \ell \leq 7$.

Thus, for example, $S1_{E16} = c_{31}c_0c_1c_2c_3c_4$, where $c = c_0c_1c_3\cdots c_{63}$ is the DES_k output (without the initial and final permutations). Further, for any $i \neq j \in \mathbb{Z}_2^{48}$ and fixed $k \in \mathbb{Z}_2^{56}$, $g_{i,k}(x)$ and $g_{j,k}(x)$ are such that $S\ell_{EX}^{i,x} = S\ell_{EX}^{j,x}$ for all $x \in \mathbb{Z}_2^{48}$, $0 \leq X \leq 16$ and $0 \leq \ell \leq 7$, where $S\ell_{EX}^{q,x}$ is the value of $S\ell_{EX}$ for $g_{q,k}(x)$ and $q \in \mathbb{Z}_2^{48}$.

On the other hand, let $x \neq y \in \mathbb{Z}_2^{64}$ and ℓ be fixed. We claim that

$$2^{-6} \leq Pr_k[S\ell_{E16}^{i,x} = S\ell_{E16}^{i,y}] < 2^{-5}$$

where the probability is taken over all keys $k \in \mathbb{Z}_2^{56}$ and $x \neq y \in \mathbb{Z}_2^{64}$.

Let A be a PRF adversary that attempts to distinguish the DES algorithm from a random function from 64 bits to 64 bits by checking whether $S\ell_{E16}^{i,x} = S\ell_{E16}^{i,y}$ for some fixed $\ell \in \{0, .., 7\}$. Then we have that

$$Adv_{DES}^{prf}(A) = Pr_k[S\ell_{E16}^{i,x} = S\ell_{E16}^{i,y}] - 2^{-6},$$

where A runs in time $t = O(56 + 64 + 64 + T_{DES})$ - T_{DES} being the time for a single execution of the DES encryption function - and makes 2 oracle queries.

It is clear that $Pr_k[S\ell_{E16}^{i,x} = S\ell_{E16}^{i,y}] \geq 2^{-6}$ (since $|S\ell_{EX}| = 6$). For the second inequality, we use the fact that

$$\begin{aligned} Adv_{DES}^{prf}(A) &\leq Adv_{DES}^{prf}(2,t) \\ &\leq c_1.\frac{t/T_{DES}}{2^{55}} + c_2.\frac{2}{2^{40}} + \frac{2}{2^{65}} \\ &< 2^{-6} \end{aligned}$$

where $c_1, c_2 \in \{0, 1\}$ (i.e. using the fact that non-key-recovery PRF distinguishers for the DES with better advantage are not yet known and that the best known key recovery attacks against the DES are either exhaustive search or linear cryptanalysis - the last term is due to the birthday paradox) and $Adv_{DES}^{prf}(2,t)$ is the PRF advantage for the adversary of maximal PRF advantage that runs in time at most t and makes at most 2 oracle queries [9].

Therefore,

$$2^{-6} \leq Pr_k[S\ell_{E16}^{i,x} = S\ell_{E16}^{i,y}] < 2^{-5}$$

as required [1, 9].

Thus, it is clear (since the DES S-box allows transitions from non-zero input XOR to zero output XOR) that for any random $i \neq j \in I$ and fixed $k \in \mathbb{Z}_2^{56}$

$$\begin{aligned} Pr\left[f_i(k) = f_j(k)\right] &\leq \sum_{\ell=1}^{8} Pr\left[wt_6(i \oplus j) = \ell\right] p^{\ell} \\ &\leq \sum_{\ell=1}^{8} \binom{8}{\ell} \frac{(2^6-1)^{\ell}}{2^{48}} p^{\ell} \\ &\leq 2^{-42} \end{aligned}$$

where $wt_6(x)$ is the six-bit weight of $x \in \mathbb{Z}_2^{48}$,

$$\begin{aligned} p &= (1-2^{-5})^6 \cdot 2^{-8} + 12 \cdot (1-2^{-5})^3 \cdot 2^{-5} \cdot 2^{-6} + 12 \cdot (1-2^{-5}) \cdot 2^{-5\times 2} \cdot 2^{-4} \\ &\quad +12 \cdot (1-2^{-5})^2 \cdot 2^{-5\times 2} \cdot 2^{-4} + 2^{-5\times 3} \cdot 2^{-2}. \end{aligned}$$

Therefore, f_i non-trivially depends on i. Further, for any two random salt values $i, j \in \mathbb{Z}_2^{48}$,

$$\begin{aligned} Pr[wt_6(i \oplus j) = 8] &= Pr[i \oplus j = Y, wt_6(Y) = 8] \\ &= \frac{(2^6-1)^8}{2^{48}} \\ &\geq 2^{-0.2}. \end{aligned}$$

Thus, it is highly likely that any two random values $i, j \in \mathbb{Z}_2^{48}$ will have a six-bit weight of 8. Therefore, given a *zero key XOR difference*, a related-key attack involving any two function instances f_i and f_j will with very high probability have eight active S-boxes in the final round of each pair $(g_{i,k}(x), g_{j,k}(x))$, $x \in \{0, 1, 2, 3\} \subset \mathbb{Z}_2^{64}$.

Moreover, we expect that for fixed $i \in \mathbb{Z}_2^{48}$, $\ell \in \{0, 1, 2, \cdots, 7\}$ and key $k \in \mathbb{Z}_2^{56}$, with probability at most $(1-2^{-5})^{6*8} = (1-2^{-5})^{48}$ the six-bit words $S\ell_{E16}^{i,0}$, $S\ell_{E16}^{i,1}$, $S\ell_{E16}^{i,2}$ and $S\ell_{E16}^{i,3}$ will be pair wise distinct. Further, the probability that the probability 1 one round differential $(\Delta S\ell_{S16}, \Delta S\ell_{O16})$ holds for any 6-bit candidate key over 4 distinct pairs is at most 2^{-8} [3]. Thus, with probability at least $1-(2^6-1)*2^{-8} = (1-2^{-2}+2^{-8}) = 0.75390625$, no random six-bit key (other than $S\ell_{K16}$) matches all the pairs.

Therefore, we expect that with probability at least 0.75390625, 1 pair of six-bit candidate keys of the form $(S\ell_{K16}, S\ell_{K16} \oplus \Delta S\ell_{S16})$ will be suggested per S-box (these may not be distinguishable since a constant input XOR value is used). Thus, with probability at least 0.75390625^8, we remain with 16 bits of entropy which we can easily recover by exhaustive search [3].

Hence, there exists a related-key PPT algorithm A such that

$$\begin{aligned} Pr\Big[f_i(z) = y : i \xleftarrow{\$} \mathbb{Z}_2^{48}, j \xleftarrow{\$} \mathbb{Z}_2^{48}; x \xleftarrow{\$} \mathbb{Z}_2^{56}; y \leftarrow f_i(x), \\ y' \leftarrow f_j(x); z \leftarrow A(i, j, y, y')\Big] &> p \cdot 0.75390625^8 \cdot q \\ &\geq 2^{-5.7} \end{aligned}$$

where $p = Pr[Wt_6(i \oplus j) = 8]$, $q = (1-2^{-5})^{48}$ and some coin tosses of A. In particular, the probability is taken over $i \in \mathbb{Z}_2^{48}$, $x \in \mathbb{Z}_2^{56}$ and some coin tosses of A. Therefore, f_c^*, is not a collection of strong one-way functions under related-I attacks which contradicts our initial hypothesis. □

Unfortunately, the above example is not merely academic - there exist security models within the password hashing domain which make the above attack practical. For example, both [26] and [21] essentially define a secure password hashing function as one which is *"as good as the passwords users choose"*. Therefore, the user responsibility is limited to choosing and storing secure passwords.

However, on the other hand, implementations that generate new salt values at each password change (e.g. the Ubuntu 12.04 LTS desktop) create, with high probability, pairs of salt values with the same password. This is more so with security policies enforcing mandatory periodic password changes which forces [many] users to keep a small set of interchangeable passwords to choose from at each mandatory expiry period (or risk - with time - generating hard to remember passwords).

Furthermore, many cross subscribing users practice cross site password reuse especially for systems with similar functionality e.g. social networking sites, web based email services, online gaming sites etc [21]. An adversary, with access to multiple password files from a collection of systems with similar functionality inverts f_c^* with high probability for each cross subscribing user regardless of the strength of the password used.

In particular, whereas password ageing [7] may ameliorate the impact of the first, the second can only be mitigated by user education with no realistic enforcement mechanism or assurance of compliance. Thus, contradicting our hypothesis on the security of the password hashing function. Therefore, f_c^* is not a secure password hashing function under this security model.

Finally, hardware and/or software optimisation on a sliding time scale (e.g. every 18 months for hardware optimisation) may account for a dramatic reduction in the adversarial inversion probability. For example, by Moore's law, the availability of faster, cheaper and smaller hardware every 18 months halves the area-time cost for inversion circuits. This results in a significant reduction in the cost of building special purpose brute force machines and/or significant increment in the amount of computational power available to parallel computing [9, 23, 21, 20].

Therefore, we require the following final postulate [21, 13, 20]

[RQ4a] the password hashing function F must depend on a configurable *time* parameter based on some elements of key stretching to ensure the adversarial inversion probability remains constant with increasing computational power and/or algorithm optimisation.

[RQ4b] the password hashing function F must include hardware frustrating techniques such as memory and/or expensive operations for imposing cost constraints on custom circuits while ensuring efficiency of computation on general purpose processors.

This leads to the following characterisation of password hashing functions.

Claim 1.2. *Any password hashing function is a collection of strong one-way functions.*

Proof. Let F be a password hashing function, then [RQ1a], [RQ1b], [RQ2], [RQ3], [RQ4a] and [RQ4b] imply that F is a strong one-way function of the form

$$F : \mathbb{N} \times T \times S \times P_A \to Ran(F)$$

where S is the salt space, $T \subseteq \mathbb{N}$ is the set of time parameters and $\mathbb{N}$ the set of natural numbers is the set of password length parameters.

The definition of F over all possible password lengths $\mathbb{N}$ reflects the standard practice for defining strong one-way functions where security is claimed asymptotically - as $k \in \mathbb{N}$ becomes larger (i.e. $k \geq MIN_p$) and the fact that F may include instructions for *padding and/or truncating* inputs (i.e. F uses equivalence classes in $\Sigma^*/\simeq_{MAX_p}$). Note that in practice, password lengths are often encoded within the password string itself (e.g. using null termination of strings etc.) we only write it explicitly for clarity.

Therefore, F is a four dimensional array indexed by a password length parameter, a time parameter, a salt value and a password value [9]. Let S_1 be an injection of the form

$$S_1 : \mathbb{N} \times T \times S \to \mathbb{N},$$

and define $F^{'} : \mathbb{N} \times P_A \to Ran(F)$ by

$$F^{'}(i,p) = \begin{cases} F(n,t,s,p) & \text{if } S_1^{-1}(i) = (n,t,s) \in \mathbb{N} \times T \times S \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$I = \left\{ i \in \mathbb{N} : S_1^{-1}(i) = (n,t,s) \in \mathbb{N} \times T \times S \right\},$$

then S_1 induces a bijection between F and $F^{''} : I \times P_A \to Ran(F)$ defined by

$$F^{''}(i,p) = F(S_1^{-1}(i), p)$$

for all $i \in I$, $p \in P_A$, $n \in \mathbb{N}$, $t \in T$ and $s \in S$.

It is clear that the password p may need some extra transformation depending on $n \in \mathbb{N}$ and $s \in S$ such as padding, truncation etc. Thus, we may write

$$F^{'''}(i, S_2(i,p)) = F^{''}(i,p) = F(S_1^{-1}(i), p)$$

for some $F^{'''}$ such that $F^{''} = F^{'''} \circ (P_1^2, S_2) : I \times P_A \to Ran(F)$ where P_1^2 is the projective map and the parameter $i \in I$ for S_2 serves to encode information about $(n,t,s) \in \mathbb{N} \times T \times S$ (through S_1^{-1}) [8]. Note that $F^{''}$ and $F^{'''}$ only differ by the initial password transformation code in S_2. For example, in addition to padding and/or truncation, if the underlying cryptographic primitive is a symmetric key cipher, this may include length based key scheduling instructions i.e. the output of S_2 may simply be the key schedule for some symmetric key cipher implemented in $F^{'''}$.

Clearly, S_1 and S_2 are PPT algorithms taking on coin tosses $(t,s) \in T \times S$ and $p \in \Sigma^*/\simeq_{MAX_p}$ respectively or F is not polynomial time computable.

Therefore, $F^{'''}$ is a collection of strong one-way functions. $\square$

In this paper, a new password hashing function, M3dcrypt is proposed. The rest of the paper is organised as follows. Section 2 discusses various background and preliminary material, Section 3 provides a detailed specification of the function, Section 4 analyses the security of the function and Section 5 explores some implementation issues.

2 Preliminaries

2.1 Notation and Other Issues

The M3dcrypt password hashing function assumes little endian byte ordering. However, big endian byte ordering can also be used so long consistency is ensured for all functions and constants [22].

Further, the M3dcrypt password hashing function assumes the natural bijection between vector spaces $(\mathbb{Z}_2^n)^m$ and $\mathbb{Z}_2^{nm}$, where $\mathbb{Z}_q$ is the set of integers modulo $q \in \mathbb{N}$. Therefore, for brevity, references to these spaces are used interchangeably. In particular, the reader will find references to elements $(\alpha_0, \alpha_1, \alpha_2 \cdots, \alpha_{m-1}) \in \mathbb{Z}_2^{nm}$, $\alpha_j \in \mathbb{Z}_2^n$ for all $0 \leq j \leq m-1$, rather than the cannonical representation $(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{m-1}) \in (\mathbb{Z}_2^n)^m$.

2.2 The M3dcrypt Key Schedule

The M3dcrypt password hashing function is based on the Advanced Encryption Standard (AES) algorithm [18, 6]. In particular, M3dcrypt implements a set of AES-like permutations

$$\mathcal{E}_{Nr} : \mathbb{Z}_2^{128(Nr+1)} \times \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$$

defined by

$$\begin{aligned} \mathcal{E}_{Nr}(Y, x) &= \sigma_{Nr} \circ \tau \circ \gamma \circ \left(\bigcirc_{i=1}^{Nr-1} \sigma_i \circ \theta \circ \tau \circ \gamma\right) \circ \sigma_0(x) \\ &= \mathcal{E}_{Nr,Y}(x) \end{aligned}$$

for all $Y \in \mathbb{Z}_2^{128(Nr+1)}$ and $x \in \mathbb{Z}_2^{128}$, where $\theta(state) = MixColumn(state)$, $\sigma_k(state) = AddRoundKey(state, Y_k)$, $\gamma(state) = ByteSub(state)$ and $\tau(state) = ShiftRow(state)$ [6, 25].

We require the following.

Let $g : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ be a fixed permutation, define the domain extension, $\hat{g}^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$, by

$$\hat{g}^m(x) = (g_0(x), g_1(x), g_2(x), \cdots, g_{m-1}(x))$$

where $x = (x_0, x_1, \cdots, x_{m-1}) \in \mathbb{Z}_2^{128m}$ and each

$$g_i(x) = g(g_{i-1}(x) \oplus x_i)$$

is recursively defined by setting $g_{-1}(z) = 0, \forall z \in \mathbb{Z}_2^{128m}$.

The domain extension $\tilde{f}^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ for some fixed permutation $f : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ is similarly defined

$$\tilde{f}^m(x) = (f_0(x), f_1(x), f_2(x), \cdots, f_{m-1}(x))$$

where each

$$f_i(x) = f(f_{i+1}(x) \oplus x_i)$$

is recursively defined by setting $f_m(z) = 0, \forall z \in \mathbb{Z}_2^{128m}$.

Claim 2.1. *The domain extension $\hat{g}^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ is a permutation.*

Proof. We prove by contradiction.

Let $x = (x_0, x_1, \cdots, x_{m-1})$, $y = (y_0, y_1, \cdots, y_{m-1}) \in \mathbb{Z}_2^{128m}$ be such that $x \neq y$ and $\hat{g}^m(x) = \hat{g}^m(y)$. Then, since g is a permutation, we must have iteratively

$$g_i(x) = g_i(y) \implies x_i = y_i, \;\; 0 \leq i \leq m-1$$

contradicting $x \neq y$. Therefore, by the size of the co-domain, $\hat{g}^m$ is a permutation. □

Claim 2.2. *The domain extension $\tilde{f}^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ is a permutation.*

Proof. Similar to Claim 2.1. □

Let $\vartheta_{Nr} : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128(Nr+1)}$ denote the Nr round AES128 key schedule and

$$\Psi = \theta \circ \tau \circ \gamma$$

denote the unkeyed AES round function.

Let $\pi_i : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ be defined by

$$\pi_i(x) = \begin{cases} \hat{g}^m(x) & \text{if } i \in \{0, 2, 4, \cdots, \}, \\ \tilde{f}^m(x) & \text{if } i \in \{1, 3, 5, \cdots, \}, \end{cases}$$

for all $x \in \mathbb{Z}_2^{128m}$ and let $\pi^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ denote the permutation defined by

$$\pi^m(x) = \bigcirc_{i=0}^{2m-1} \pi_i(x)$$

for all $x \in \mathbb{Z}_2^{128m}$.

Then, the M3dcrypt key schedule *key initialisation function*

$$I_m^f : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128(2m)},$$

is defined by **Algorithm 1** below.

Algorithm 1: Key Initialisation Function, I_m^f

Require: $key \in \mathbb{Z}_2^{128m}$.

$I_m^f(key)$**:**
  $(k_0, k_1, \cdots, k_{m-1}) := (\pi^m(key) \oplus key)$
  $(k_m, k_{m+1}, \cdots, k_{2m-1}) := \pi^m(key)$
  for $i := 0$ **to** $m-1$ **do**
    $k_i := \Psi(k_i)$
  end for

  Return $(k_0, k_1, \cdots, k_{m-1}, k_m, k_{m+1}, \cdots, k_{2m-1})$

The M3dcrypt key schedule *key extraction function*

$$f_m^X : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128(N_r-2m+1)}$$

is defined by **Algorithm 2** below.

Algorithm 2: Key Extraction Function, f_m^X

Require: $(k_0, k_1, \cdots, k_{m-1}) \in \mathbb{Z}_2^{128m}$

$f_m^X(k_0, k_1, \cdots, k_{m-1})$**:**
  $p := 0$; $\omega_{-1} := 0$; $\phi_{-1} := 0$
  while $p < (Nr - 2m + 1)$ **do**
    $\omega_p := \Psi\left(\omega_{p-1} \oplus (p+1) \oplus \left(\bigoplus_{i=p}^{p+m-1} k_i\right)\right)$
    $k_{p+m} := \Psi(\phi_{p-1} \oplus \omega_p)$
    $\phi_p := \Psi(\phi_{p-1} \oplus k_{p+m})$
    $p := p + 1$
  end while

  Return $(k_m, k_{m+1}, k_{m+2}, \cdots, k_{Nr-m})$

Finally, for $m > 1$, define $\varphi_{Nr}^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128(Nr+1)}$ the Nr-round M3dcrypt key schedule for $128m$-bit master keys by

$$\varphi_{Nr}^m(key) \quad = \quad rROT_{128m}\left(I_m^f(key), f_m^X(\pi^m(key))\right)$$

where $key \in \mathbb{Z}_2^{128m}$ and $rROT_k$ is the k-bit *right cyclic shift* function.

2.3 The M3dcrypt Constants

The M3dcrypt constants are based on the AES128 key schedule for master key 0, $\vartheta_{Nr}(0)$, where Nr is the number of AES128 rounds. Therefore, for example, $C = \vartheta_3(0)$ is defined by

$$
\begin{aligned}
C_0 &= \{0x00000000, 0x00000000, 0x00000000, 0x00000000\},\\
C_1 &= \{0x63636362, 0x63636362, 0x63636362, 0x63636362\},\\
C_2 &= \{0xc998989b, 0xaafbfbf9, 0xc998989b, 0xaafbfbf9\},\\
C_3 &= \{0x50349790, 0xfacf6c69, 0x3357f4f2, 0x99ac0f0b\},
\end{aligned}
$$

in hexadecimal.

2.4 Properties of the M3dcrypt Key Schedule

Claim 2.3. *For $Nr \geq 3m$ $(m > 1)$, pairs of equivalent keys in φ^m_{Nr} are unlikely.*

Proof. Pairs of equivalent keys are a certainty if there exist pairs of keys $key_0 \neq key_1 \in \mathbb{Z}_2^{128m}$ such that $\varphi^m_{Nr}(key_0) = \varphi^m_{Nr}(key_1)$.

Since $\pi^m(key_0) \neq \pi^m(key_1)$ is part of the subkey sequence whenever $Nr \geq 3m$, $\varphi^m_{Nr}(key_0) \neq \varphi^m_{Nr}(key_1)$ for all $key_0 \neq key_1 \in \mathbb{Z}_2^{128m}$ and $Nr \geq 3m$. □

Claim 2.4. *For $Nr \geq 3m$ $(m > 1)$, related-key differential attacks in φ^m_{Nr} are unlikely.*

Proof. Related-key attacks exist in ciphers in which an adversary is able to simultaneously transition non-trivial differences through both the key schedule and the cipher inner state.

Since, on average, a brute force attack requires 2^{n-1} rekeyings [24, 22], any n-bit key schedule in which transitioning non-trivial differences has maximum probability 2^{1-n} is resilient against the attack, otherwise $O(2^{n-1})$ is polynomial.

However, in effect, this merely re-states the requirement for key schedule resilience against differential attacks [14, 5].

On the other hand, bearing in mind the arguments of [19, 3], we note that π^m has differential propagation ratio at most $2^{-120(2m+1)}$. Hence, resistance against related-key attacks holds whenever the following inequality holds

$$240m + 120 \quad > \quad 128m - 1$$

and thus whenever $m > 1$ (being our base assumption). □

3 The M3dcrypt Password Hashing Algorithm

Let $S = \mathbb{Z}_2^{256}$ be the salt space, $P_A \subseteq \mathbb{Z}_2^{384}$ be the password space (depending on the prevailing security policy) and $T = T_0 \times T_1 \subset \mathbb{Z}_{2^{32}} \times \mathbb{Z}_{2^{32}}$ be the set of time parameters, where $T_0 = \{2^j : 20 \leq j < 32\}$, $T_1 = \{j : 1 \leq j < 2^{32}\}$ and for any time parameter $(t_0, t_1) \in T$, the time complexity for the M3dcrypt password hashing function F is some function $\delta(t_0, t_1)$ of both $t_0 \in T_0$ and $t_1 \in T_1$ while its space (memory) complexity is a function of $t_0 \in T_0$.

Then, the M3dcrypt password hashing function, F is the collection of functions defined as follows [9].

Let $S_1 : \mathbb{Z}_{49} \times T \times S \to \mathbb{Z}_{2^{326}}$ denote the injection defined by

$$S_1(n, t, s) = 2^{320}n + 2^{288}t_1 + 2^{256}t_0 + s,$$

where $t = (t_0, t_1) \in T$, n is the password length in bytes and $s \in S$, define

$$I = \{\, 2^{320}n + 2^{288}t_1 + 2^{256}t_0 + s : \; n \in \mathbb{Z}_{49}, \; (t_0, t_1) \in T, \; s \in S \,\}.$$

For any $t = (t_0, t_1) \in T$, define

$$tcost = \tfrac{t_0}{8}, \; lm = \log_2(tcost), \; skey = (\text{0xd09788fd} \,\%\, tcost),$$

then, for any $lm \in \{j + 17 : j \in \mathbb{Z}_{15}\}$ define ζ_{lm} by

lm	ζ_{lm}		**lm**	ζ_{lm}		**lm**	ζ_{lm}
17	29		22	15		27	29
18	7		23	11		28	3
19	21		24	43		29	11
20	7		25	35		30	3
21	17		26	15		31	15

Let $\lambda : \mathbb{Z}_{tcost} \to \mathbb{Z}_{t_0}$ be the injection defined by

$$\lambda(x) \quad = \quad (l \circ \psi \circ \upsilon(x)) << 3,$$

where

$$\begin{aligned} \upsilon(x) &= ((\text{0xfc6564bd} + x) * \zeta_{lm}) \,\%\, tcost, \\ \psi(y) &= rROT_3(y) \oplus skey, \\ l(z) &= rROT_1(z) \oplus lROT_8(z) \oplus lROT_{15}(z), \end{aligned}$$

$x, y, z \in \mathbb{Z}_{tcost}$, $rROT_j$ and $lROT_j$ are the right and left cyclic shifts of an lm-bit input by j bits respectively.

Finally, let $S_2 : I \times P_A \to \mathbb{Z}_2^{384}$ denote the function defined by **Algorithm 3** below and $A_1 : I \times \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{512}$ be defined by

$$\begin{aligned} A_1(i, x) &= f_i(x) \\ &= \left(\mathcal{E}_{20,\varphi_{20}^3(x)}(C_0), \mathcal{E}_{20,\varphi_{20}^3(x)}(C_1), \mathcal{E}_{20,\varphi_{20}^3(x)}(C_2), \mathcal{E}_{20,\varphi_{20}^3(x)}(C_3)\right), \end{aligned}$$

for all $i \in I$ and $x \in \mathbb{Z}_2^{384}$.

Then, the M3dcrypt password hashing function F is the collection of functions

$$F = \{f_i : \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{512}\}_{i \in I}.$$

Algorithm 3: $S_2 : I \times P_A \to \mathbb{Z}_2^{384}$

Require: $i \in I$, $p \in P_A \subseteq \mathbb{Z}_2^{384}$

$S_2(i,p)$**:**

$$\begin{array}{lllllll}
t_0 & := & (i >> 256)\ \&\ \text{0xffff}, & t_1 & := & (i >> 288)\ \&\ \text{0xffff}, \\
n & := & (i >> 320)\ \&\ \text{0x3f}, & tcost & := & \frac{t_0}{8}, \\
s & := & i\ \&\ (2^{256}-1), & key & := & (p||0^{384-8n}), \\
v & := & (0,0,0) \in \mathbb{Z}_2^{384}, & u & := & (0^{64}||t_0||t_1)
\end{array}$$

for $z := 0$ **to** 3 **do**
  $X_z := \mathcal{E}_{4,\vartheta_4(0)}\left(z \oplus \mathcal{E}_{20,\varphi_{20}^3(key)}\left(\mathcal{E}_{20,\varphi_{20}^2(s)}(C_z \oplus u)\right)\right)$
end for
for $z := 4$ **to** $t_0 - 1$ **do**
  $X_z := \mathcal{E}_4\left(\vartheta_4(0), X_{z-1} \oplus X_{z-4} \oplus z\right)$
end for

$tkey_0 := (X_{t_0-8}, X_{t_0-7}, X_{t_0-6}, \cdots, X_{t_0-1})$
$tkey_1 := (X_0, X_1, X_2)$

$v := \tilde{\mathcal{E}}^3_{20,\varphi_{20}^3(tkey_1)}\left(\hat{\mathcal{E}}^3_{7,tkey_0}(v)\right)$
for $j := 0$ **to** $t_1 - 1$ **do**
  $ts := 0, \quad tkey := (0,0,0,0,0,0,0,0)$
  for $z := 0$ **to** $tcost - 1$ **do**
    $q := \lambda(z)$
    $tkey := (X_q, X_{q+1}, X_{q+2}, \cdots, X_{q+7})$
    $ts := z \,\%\, 6$
    if $ts < 3$ **then**
      if $ts = 0$ **then**
        $v_0 := \mathcal{E}_7(tkey, v_0 \oplus j)$
      else
        $v_{ts} := \mathcal{E}_7(tkey, v_{ts} \oplus v_{ts-1})$
      end if
    else
      if $ts = 3$ **then**
        $v_2 := \mathcal{E}_7(tkey, v_2 \oplus j)$
      else
        $v_{5-ts} := \mathcal{E}_7(tkey, v_{5-ts} \oplus v_{6-ts})$
      end if
    end if
  end for
  $do\_finish(v, tkey, ts, tcost)$
end for
$v := \hat{\mathcal{E}}^3_{20,\varphi_{20}^3(key)}(v)$

Return v

where the function *do_finish* is given by **Algorithm 4** below.

Algorithm 4: *do_finish*( )

Require: $v \in \mathbb{Z}_2^{384}$, $tkey \in \mathbb{Z}_2^{1024}$, $ts \in \mathbb{Z}_6$, $tcost = \frac{t_0}{8}$

```
do_finish(v, tkey, ts, tcost):
  if ts < 3 then
    if (tcost % 3) = 2 then
      v_2 := E_7(tkey, v_2 ⊕ v_1)
    else
      v_1 := E_7(tkey, v_1 ⊕ v_0)
      v_2 := E_7(tkey, v_2 ⊕ v_1)
    end if
  else
    if (tcost % 3) = 2 then
      v_0 := E_7(tkey, v_0 ⊕ v_1)
    else
      v_1 := E_7(tkey, v_1 ⊕ v_2)
      v_0 := E_7(tkey, v_0 ⊕ v_1)
    end if
  end if
```

4 Security Analysis

For this section, we require the following property (Claim 4.1) of [pseudo]random permutations.

Claim 4.1. *For any two distinct permutations $\rho_0, \rho_1 : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ and any two elements $x, y \in \mathbb{Z}_2^n$,*

$$Pr[\rho_0(x) = \rho_1(y)] = \begin{cases} 2^{-n} & \text{if } \rho_0 \neq \rho_1 \\ 1 & \text{if } \rho_0 = \rho_1 \text{ and } x = y \\ 0 & \text{if } \rho_0 = \rho_1 \text{ and } x \neq y \end{cases}$$

Proof. Since the second and last cases are clear, we consider the case $\rho_0 \neq \rho_1$. We have,

$$\begin{aligned} Pr[\rho_0(x) = \rho_1(y)] &= \sum_{z \in \mathbb{Z}_2^n} Pr[\rho_0(x) = z | \rho_1(y) = z] \cdot Pr[\rho_1(y) = z] \\ &= 2^n \cdot \frac{1}{2^{2n}} \\ &= 2^{-n} \end{aligned}$$

□

4.1 Properties of the X Array

Claim 4.2. *For any fixed random password and $i \in I$, any set of six consecutive elements of the X array has at least two distinct elements.*

Proof. We prove by contradiction.

Let $X_{z-4} = X_{z-3} = X_{z-2} = \cdots = X_z = X_{z+1}$, $(4 \leq z \leq t_0 - 2)$ be a set of 6 consecutive elements of the X array for a fixed random password and $i \in I$.

Then we must have

$$\begin{aligned} \mathcal{E}_4\left(\vartheta_4(0), X_{z-4} \oplus X_{z-1} \oplus z\right) &= X_z \\ &= X_{z+1} \\ &= \mathcal{E}_4\left(\vartheta_4(0), X_{z-3} \oplus X_z \oplus (z+1)\right). \end{aligned}$$

Since $X_{z-4} = X_{z-1}$ and $X_{z-3} = X_z$, we have a contradiction. $\square$

Claim 4.3. *For any fixed random password and $i \in I$, there are with high probability at least two distinct 128-bit elements in every set of five elements of the X array.*

Proof. For brevity, we abuse notation as follows.

Let $p \in P_A$, $s \in S$, $(t_0, t_1) \in T$ be fixed, let $key = (p||0^{384-|p|}) \in \mathbb{Z}_2^{384}$ and $u = (0^{64}||t_0||t_1)$, set $X_{z-4} = \mathcal{E}_{20,\varphi_{20}^3(key)}\left(\mathcal{E}_{20,\varphi_{20}^2(s)}(C_z \oplus u)\right)$, $0 \leq z \leq 3$. Further, for $0 \leq k \leq t_0 - 1$ define

$$X_{k-1}^* = \begin{cases} 0 & \text{if } 0 \leq k \leq 3 \\ X_{k-1} & \text{if } 4 \leq k \leq t_0 - 1. \end{cases}$$

Therefore, for any $0 \leq z \neq j \leq t_0 - 1$

$$\begin{aligned} \mathcal{E}_4\left(\vartheta_4(0), X_{z-1}^* \oplus X_{z-4} \oplus z\right) &= X_z \\ &= X_j \\ &= \mathcal{E}_4\left(\vartheta_4(0), X_{j-1}^* \oplus X_{j-4} \oplus j\right) \end{aligned}$$

implies $(X_{z-1}^* \oplus X_{z-4}) \oplus (X_{j-1}^* \oplus X_{j-4}) = z \oplus j$.

Since z and j are fixed integers, we have

$$Pr\left[(X_{z-1}^* \oplus X_{z-4}) \oplus (X_{j-1}^* \oplus X_{j-4}) = z \oplus j\right] = 2^{-128}.$$

Therefore, all elements of any 5 element set of the X array are equal with probability at most

$$\begin{aligned} \binom{2^{31}}{5} 2^{-128 \cdot \binom{5}{2}} &< 2^{155-1280} \\ &= 2^{-1125}. \end{aligned}$$

Therefore, with probability at least $1-2^{-1125}$ there are at least two distinct 128-bit elements in any set of five elements from the X array. $\square$

In a way the above proof carries inherent risk in assuming that $\mathcal{E}_{4,\vartheta_4(0)}$ acts randomly on its inputs (which may be a far stronger assumption in reality). However, all that is required for the proof is that elements of the X-array are not designed for particular relationships to hold with high probability.

In particular, it is clear that the output of $\mathcal{E}_{4,\vartheta_4(0)}$ at any single point i is independent of that at any other point $j \neq i$ (for any two fixed integers $0 \leq i \neq j \leq t_0 - 1$) which ensures that the above probabilities hold regardless of our assumptions on $\mathcal{E}_{4,\vartheta_4(0)}$.

Note that the idea of including iteration count j in the computation of the j^{th} output of an iteratively computed value so as to increase the resultant entropy is well known, for example see [16, 27, 11].

Claim 4.4. *For any fixed random password and $i \in I$, the X array is not composed of a single repeating cycle of length greater than four.*

Proof. We prove by contradiction.

By definition X has a cycle if we can find ℓ, $0 \leq \ell \leq t_0 - \mu - 1$ and $\mu > 1$ such that there exists a leading sequence $X_0, X_1, \cdots, X_{\ell-1}$ called a *leader* and a *cycle* $X_\ell, X_{\ell+1}, \cdots, X_{\ell+\mu-1}$ of length μ such that $X_\ell = X_{\ell+\mu}$ [12].

Suppose $X = \{X_0, X_1, \cdots, X_{\mu-1}, X_0, X_1, \cdots, X_{\mu-1}, \cdots\}$ for some $\mu > 1$. Consider any two points z and j in distinct cycles such that $X_{z+k} = X_{j+k}$, $0 \leq k \leq 4$, $0 \leq z \neq j \leq t_0 - 1$. We must have

$$\begin{aligned}
\mathcal{E}_4\left(\vartheta_4(0), X_{z+3} \oplus X_z \oplus (z+4)\right) &= X_{z+4} \\
&= X_{j+4} \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{j+3} \oplus X_j \oplus (j+4)\right)
\end{aligned}$$

Since $X_{z+3} = X_{j+3}$ and $X_z = X_j$ we have a contradiction for $\mathcal{E}_{4,\vartheta_4(0)}$.

Therefore, we must have $\mu \leq 4$. $\square$

In effect, Claim 4.4 proves a stronger result i.e. that X does not contain *any* repeated sequence of length more than 4. This leads to Claim 4.5.

Claim 4.5. *For any fixed random password and $i \in I$, the X array is not composed of any single repeating cycle.*

Proof. We prove by contradiction.

Suppose $X = \{X_0, X_1, \cdots, X_{\mu-1}, X_0, X_1, \cdots, X_{\mu-1}, \cdots\}$ for some $\mu \in \{2, 3, 4\}$ by Claim 4.4.

Then, if $\mu = 2$, we must have $X_z = X_{z-2}$ for all $2 \leq z \leq t_0 - 1$ and therefore,

$$\begin{aligned}
\mathcal{E}_4\left(\vartheta_4(0), X_{z+3} \oplus X_z \oplus (z+4)\right) &= X_{z+4} \\
&= X_{z+6} \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+5} \oplus X_{z+2} \oplus (z+6)\right) \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+3} \oplus X_z \oplus (z+6)\right),
\end{aligned}$$

a contradiction.

On the other hand, if $\mu = 3$, $X_z = X_{z-3}$ must hold for all $3 \leq z \leq t_0 - 1$ and therefore,

$$\begin{aligned}
\mathcal{E}_4\left(\vartheta_4(0), X_{z+4} \oplus X_{z+1} \oplus (z+5)\right) &= X_{z+5} \\
&= X_{z+8} \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+7} \oplus X_{z+4} \oplus (z+8)\right) \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+4} \oplus X_{z+1} \oplus (z+8)\right),
\end{aligned}$$

a contradiction.

Finally, if $\mu = 4$, we must have $X_z = X_{z-4}$ for all $4 \leq z \leq t_0 - 1$ and therefore,

$$\begin{aligned}
\mathcal{E}_4\left(\vartheta_4(0), X_{z+3} \oplus X_z \oplus (z+4)\right) &= X_{z+4} \\
&= X_{z+8} \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+7} \oplus X_{z+4} \oplus (z+8)\right) \\
&= \mathcal{E}_4\left(\vartheta_4(0), X_{z+3} \oplus X_z \oplus (z+8)\right),
\end{aligned}$$

a contradiction. □

As it turns out, we can prove a stronger result.

Claim 4.6. *For any fixed random password and $i \in I$, the probability that X has n distinct non-overlapping pairs of repeated sequences of length $\ell \geq 2$ is bounded by $2^{(62-128\ell)n}$.*

Proof. First, we adopt notation from Claim 4.3 as follows. For any $0 \leq k \leq t_0 - 1$ define

$$X^*_{k-1} = \begin{cases} 0 & \text{if } 0 \leq k \leq 3 \\ X_{k-1} & \text{if } 4 \leq k \leq t_0 - 1. \end{cases}$$

Then for any repeated sequence,

$$\{X_z, X_{z+1}, \cdots, X_{z+\ell-1}\} = \{X_j, X_{j+1}, \cdots, X_{j+\ell-1}\} \subset X,$$

$0 \leq z \neq j \leq t_0 - \ell$ and $2 \leq \ell \leq 4$, we have that

$$\begin{aligned}
\mathcal{E}_{4,\vartheta_4(0)}\left(X^*_{z-1+\mu} \oplus X_{z-4+\mu} \oplus (z+\mu)\right) &= X_{z+\mu} \\
&= X_{j+\mu} \\
&= \mathcal{E}_{4,\vartheta_4(0)}\left(X^*_{j-1+\mu} \oplus X_{j-4+\mu} \oplus (j+\mu)\right)
\end{aligned}$$

for all $0 \le \mu \le \ell - 1$ implies $(X^*_{z-1+\mu} \oplus X_{z-4+\mu}) \oplus (X^*_{j-1+\mu} \oplus X_{j-4+\mu}) = (z+\mu) \oplus (j+\mu)$ for all $0 \le \mu \le \ell - 1$.

Since i, z and μ are fixed integers and assuming $\mathcal{E}_{4,\vartheta_4(0)}$ acts randomly on its inputs, any such repeated sequence occurs with probability

$$Pr\left[(X_{z-1+\mu} \oplus X_{z-4+\mu}) \oplus (X_{j-1+\mu} \oplus X_{j-4+\mu}) = (z+\mu) \oplus (j+\mu)\right]^{\ell} = 2^{-128\ell}.$$

Therefore, the probability that any given n distinct pairs of X-array indices represent starting points of n distinct pairs of repeated sequences of length ℓ is

$$2^{-128\ell n}.$$

However, the X array is not demarcated using ℓ length vectors of 128-bit elements. Therefore, any $0 \le z \le t_0 - \ell$ is a possible sequence starting point. Hence, there are at most

$$\binom{2^{31}}{2n} < 2^{62n}$$

possible (overlapping) n distinct pairs of sequences of length ℓ which is the maximum number of possible ways of choosing $2n$ distinct starting points for distinct sequences of X indices.

Therefore, the probability that X has n distinct non-overlapping pairs of repeated sequences of length ℓ is bounded by $2^{62n} \cdot 2^{-128\ell n} = 2^{(62-128\ell)n}$. □

Table 2 below gives a list of probabilities for various possible (or combinations of) number (n) of repeated pairs of sequences for fixed ℓ, $2 \le \ell \le 4$, in X assuming 384-bit passwords and $|I| = 2^{298}$.

ℓ/n	**2**	**3**	**4**	**Maximum Probability**
	1	0	0	2^{-194}
	0	1	0	2^{-322}
	0	0	1	2^{-450}
	2	0	0	2^{-388}
	0	2	0	2^{-644}
	3	0	0	2^{-582}
	1	1	0	2^{-516}
	1	0	1	2^{-644}

Table 2: *Probabilities [of combinations] of repeated sequences in X.*

Therefore, more than three pairs of distinct repeated sequences are unlikely in X. This implies that the adversary acquires no non-trivial complexity gain by exploiting regularities in the X array.

Claim 4.7. *Any set of 4 consecutive elements of the X array for any two distinct passwords and fixed $i \in I$ are distinct.*

Proof. We prove by contradiction.

Suppose there exist two distinct passwords $x \neq y \in P_A$ and a fixed $i \in I$ such that

$$\{X^x_z, X^x_{z+1}, X^x_{z+2}, X^x_{z+3}\} = \{X^y_z, X^y_{z+1}, X^y_{z+2}, X^y_{z+3}\}$$

where X^v is the [ordered] X array for the password $v \in P_A$ and $z \geq 0$.

Then, we must have that

$$\begin{aligned}\mathcal{E}_4\left(\vartheta_4(0), X^x_{z+2} \oplus X^x_{z-1} \oplus (z+3)\right) &= X^x_{z+3} \\ &= X^y_{z+3} \\ &= \mathcal{E}_4\left(\vartheta_4(0), X^y_{z+2} \oplus X^y_{z-1} \oplus (z+3)\right)\end{aligned}$$

implies $X^x_{z-1} = X^y_{z-1}$. Similarly, we have $X^x_{z-2} = X^y_{z-2}$, $X^x_{z-3} = X^y_{z-3}$ and $X^x_{z-4} = X^y_{z-4}$.

Applying this iteratively, we arrive at $X^x_0 = X^y_0, X^x_1 = X^y_1, X^x_2 = X^y_2$ and $X^x_3 = X^y_3$. However, this can only happen with probability 2^{-512} which is unlikely for 384-bit passwords. □

Claim 4.8. *Any set of* 4 *consecutive elements of the* X *array for any two distinct salt values, fixed password and* T *values are distinct.*

Proof. Similar to Claim 4.7. □

4.2 Properties of S_2

Claims 4.2 - 4.6 show that the risk of adversarial complexity gain in computing X values through exploiting regularities in the array are minimal. However, the rest of S_2 especially the final two *for loops* involve the iterative application of [a composition of] two primitives defined below.

Therefore, we need to assess possibilities for adversarial complexity gain through exploiting cycles induced by the action of the composite primitive for fixed $i \in I$ and $p \in P_A$ [13, 27, 12]. We require the following.

For each $j \in J = \{j : 0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 1\}$ and $z \in \mathbb{Z}_{t_1}$, define

$$G_{z,j} = \begin{cases} \left(g^z_{\lambda(3j)}, g^0_{\lambda(3j+1)}, g^0_{\lambda(3j+2)}\right) & \text{if } 0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2, \\ \left(g^z_{\lambda(3j)}, g^0_{\lambda(3j)}, g^0_{\lambda(3j)}\right) & \text{if } j = \lfloor \frac{tcost}{3} \rfloor - 1 \text{ and } (tcost \,\%\, 3) = 1, \\ \left(g^z_{\lambda(3j)}, g^0_{\lambda(3j+1)}, g^0_{\lambda(3j+1)}\right) & \text{if } j = \lfloor \frac{tcost}{3} \rfloor - 1 \text{ and } (tcost \,\%\, 3) = 2. \end{cases}$$

where $g^z_\ell = \mathcal{E}_{7,(X_\ell \oplus z, X_{\ell+1}, X_{\ell+2}, \cdots, X_{\ell+7})}$, for all $0 \leq \ell \leq t_0 - 8$.

Further, for fixed $z \in \mathbb{Z}_{t_1}$, let $\hat{G}_z, \tilde{G}_z : J \times \mathbb{Z}^{384}_2 \to \mathbb{Z}^{384}_2$ denote the transformations induced by $G_{z,j}$ on $\mathbb{Z}^{384}_2$, for fixed $j \in J$, defined in **Table 3** below.

Conditions	$\hat{G}_z(j,x)$
$0 \le j \le \lfloor \frac{tcost}{3} \rfloor - 2$	$\left(g^z_{\lambda(3j)}(x_0), g^0_{\lambda(3j+1)}(u_0 \oplus x_1), g^0_{\lambda(3j+2)}(v_0 \oplus x_2)\right)$
$j = \lfloor \frac{tcost}{3} \rfloor - 1,\ (tcost\,\%\,3) = 1$	$\left(g^z_{\lambda(3j)}(x_0), g^0_{\lambda(3j)}(u_0 \oplus x_1), g^0_{\lambda(3j)}(v_0 \oplus x_2)\right)$
$j = \lfloor \frac{tcost}{3} \rfloor - 1,\ (tcost\,\%\,3) = 2$	$\left(g^z_{\lambda(3j)}(x_0), g^0_{\lambda(3j+1)}(u_0 \oplus x_1), g^0_{\lambda(3j+1)}(v_0 \oplus x_2)\right)$
Conditions	$\tilde{G}_z(j,x)$
$0 \le j \le \lfloor \frac{tcost}{3} \rfloor - 2$	$\left(g^0_{\lambda(3j+2)}(v_1 \oplus x_0), g^0_{\lambda(3j+1)}(u_1 \oplus x_1), g^z_{\lambda(3j)}(x_2)\right)$
$j = \lfloor \frac{tcost}{3} \rfloor - 1,\ (tcost\,\%\,3) = 1$	$\left(g^0_{\lambda(3j)}(v_1 \oplus x_0), g^0_{\lambda(3j)}(u_1 \oplus x_1), g^z_{\lambda(3j)}(x_2)\right)$
$j = \lfloor \frac{tcost}{3} \rfloor - 1,\ (tcost\,\%\,3) = 2$	$\left(g^0_{\lambda(3j+1)}(v_1 \oplus x_0), g^0_{\lambda(3j+1)}(u_1 \oplus x_1), g^z_{\lambda(3j)}(x_2)\right)$

Table 3: *Values for* $\hat{G}_z(j,x) = \hat{G}_{z,j}(x)$ *and* $\tilde{G}_z(j,x) = \tilde{G}_{z,j}(x)$ *for all* $x \in \mathbb{Z}_2^{384}$

where $x = (x_0, x_1, x_2) \in \mathbb{Z}_2^{384}$, $u_0 = g^z_{\lambda(3j)}(x_0)$, $u_1 = g^z_{\lambda(3j)}(x_2)$,

$$v_0 = \begin{cases} g^0_{\lambda(3j)}(u_0 \oplus x_1) & \text{if } j = \lfloor \frac{tcost}{3} \rfloor - 1 \text{ and } (tcost\,\%\,3) = 1, \\ g^0_{\lambda(3j+1)}(u_0 \oplus x_1) & \text{otherwise,} \end{cases}$$

and

$$v_1 = \begin{cases} g^0_{\lambda(3j)}(u_1 \oplus x_1) & \text{if } j = \lfloor \frac{tcost}{3} \rfloor - 1 \text{ and } (tcost\,\%\,3) = 1, \\ g^0_{\lambda(3j+1)}(u_1 \oplus x_1) & \text{otherwise.} \end{cases}$$

Finally, for $z \in \mathbb{Z}_{t_1}$, define $G^X_z : J \times \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{384}$ by

$$\begin{aligned} G^X_z(j,x) &= \begin{cases} \hat{G}_{z,j}(x) & \text{if } j \in \{0,2,4,6,\cdots\}, \\ \tilde{G}_{z,j}(x) & \text{if } j \in \{1,3,5,7,\cdots\}, \end{cases} \\ &= G^X_{z,j}(x), \end{aligned}$$

for all $j \in J$ and $x \in \mathbb{Z}_2^{384}$.

Then, for any $z \in \mathbb{Z}_{t_1}$, the last (or inner) *for loop* of S_2 can be written as

$$\bigcirc_{j=0}^{\lfloor \frac{tcost}{3} \rfloor - 1} G^X_{z,j}.$$

Claim 4.9. *The transformations* $\hat{G}_{z,j}, \tilde{G}_{z,j} : \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{384}$ *for fixed* $j \in J$ *and* $z \in \mathbb{Z}_{t_1}$ *are permutations.*

Proof. We prove by contradiction.

Suppose there exists $x = (x_0, x_1, x_2) \neq y = (y_0, y_1, y_2) \in \mathbb{Z}_2^{384}$ such that $\hat{G}_{z,j}(x) = \hat{G}_{z,j}(y)$, $0 \le j \le \lfloor \frac{tcost}{3} \rfloor - 2$. Then we must have that

$$\begin{aligned} g^z_{\lambda(3j)}(x_0) &= g^z_{\lambda(3j)}(y_0), \\ g^0_{\lambda(3j+1)}(u_0 \oplus x_1) &= g^0_{\lambda(3j+1)}(u_0 \oplus y_1), \\ g^0_{\lambda(3j+2)}(v_0 \oplus x_2) &= g^0_{\lambda(3j+2)}(v_0 \oplus y_2), \end{aligned}$$

which implies $x_0 = y_0$, $x_1 = y_1$ and $x_2 = y_2$, a contradiction.

Similar analysis for the case $j = \lfloor \frac{tcost}{3} \rfloor - 1$ shows that $\hat{G}_{z,j}(x) \neq \hat{G}_{z,j}(y)$ whenever $x \neq y \in \mathbb{Z}_2^{384}$ and the result follows from the size of the co-domain. The proof for $\tilde{G}_{z,j}$ is similar. $\square$

Therefore, the final two *for loops* are clearly a permutation on $\mathbb{Z}_2^{384}$.

Claim 4.10. *Repeated cycles in the last two* ***for loops*** *of S_2 are unlikely.*

Proof. From the above discussion, the last two *for loops* of S_2 can be written as

$$\bigcirc_{z=0}^{t_1-1} \left(\bigcirc_{j=0}^{\lfloor \frac{tcost}{3} \rfloor - 1} G_{z,j}^X \right).$$

There are two options. Either the adversary takes advantage of repeated cycles within $\left(\bigcirc_{j=0}^{\lfloor \frac{tcost}{3} \rfloor - 1} G_{z,j}^X \right)$ for fixed z which allows it to iteratively apply the cycles through all $z \in \mathbb{Z}_{t_1}$. Alternatively, the adversary might target a cycle over multiple $z \in \mathbb{Z}_{t_1}$ i.e. over some position(s) $(j, j^{'})$ (not necessarily distinct) in different iteration rounds say $(z, z^{'})$, $z \neq z^{'}$, which allows it to iterate the sequence through all $z \in \mathbb{Z}_{t_1}$ (or just a substantial part thereof).

For brevity, the first option requires at least one complete cycle to occur before the last run of G_z^X or the adversary acquires trivial complexity gain.

Clearly, each $\hat{G}_{z,j}$ and $\tilde{G}_{z,j}$, $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2$, is based on three distinct permutations on 128 bits (i.e. the three component functions of $G_{z,j}$) by Claim 4.4. Further, each permutation g_ℓ^z, $z \in \mathbb{Z}_{t_1}$, $0 \leq \ell \leq t_0 - 8$, has a key space of size at least

$$2^{128} \cdot (2^{128} - 1) \cdot (2^{128} - 2) \cdots (2^{128} - 7) = \frac{2^{128}!}{(2^{128} - 8)!}$$

by Claim 4.2 and Claim 4.3 (for example, one should consider the effect of moving the salt value through the entire salt space - 2^{256} elements - for fixed T [and fixed password] and applying Claim 4.8).

Therefore, we expect each $\tilde{G}_{z,j+1} \circ \hat{G}_{z,j}$ or $\hat{G}_{z,j+1} \circ \tilde{G}_{z,j}$, $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2$, to have a key space of at least

$$\left(2^{128} \cdot (2^{128} - 1) \cdot (2^{128} - 2) \cdots (2^{128} - 47)\right) \quad = \quad \left(\frac{2^{128}!}{(2^{128} - 48)!} \right),$$

since $\hat{G}_{z,j+\mu}$ and $\tilde{G}_{z,j+\mu}$, $\mu \in \{0, 1\}$ are distinct permutations for all $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2$ and $z \in \mathbb{Z}_{t_1}$.

Thus, for fixed $z \in \mathbb{Z}_{t_1}$, if either

$$\tilde{G}\hat{G} = \left\{ \tilde{G}_{z,j+1} \circ \hat{G}_{z,j} : \; 0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2 \right\},$$

or

$$\hat{G}\tilde{G} = \left\{ \hat{G}_{z,j+1} \circ \tilde{G}_{z,j} : \; 0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2 \right\},$$

(depending on the adversary's plan of attack) were a group or subgroup of an even larger group it would have size at least $\left(\frac{2^{128}!}{(2^{128}-48)!}\right)$, since z simply permutes the key space. Hence, one expects a cycle to occur with high probability after

$$2^{192} = \min\left\{2 \cdot \left(\frac{2^{128}!}{(2^{128}-48)!}\right)^{\frac{1}{2}}, 2^{\frac{384}{2}}\right\}$$

successive iterations of G_z^X [12].

However, the maximum number of iterations of G_z^X for fixed $z \in \mathbb{Z}_{t_1}$ in S_2 is only

$$\lceil \frac{tcost}{3} \rceil \quad < \quad 2^{27}.$$

In particular, assuming G_z^X (for fixed z) is a random function of its inputs, the probability of a collision in the v values is at most

$$\frac{2^{27 \cdot 2}}{2^{385}} = 2^{-331}.$$

However, after any such collisions, each successive $G_{z,j}$ the repeated v value encounters is component wise distinct by Claim 4.4, hence each repeat of previous values occurs with probability 2^{-384} by Claim 4.1. Therefore, even very short repeated sequences are unlikely using this method.

On the other hand, an adversary that targets cycles across multiple $z \in \mathbb{Z}_{t_1}$, requires a collision across those $z \in \mathbb{Z}_{t_1}$ which leads into a repeated sequence over all t_1 iterations or just a majority of them.

However, we note that for $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 2$,

$$\begin{aligned}
\hat{G}_z(j,x) &= \left(g^z_{\lambda(3j)}(x_0), g^0_{\lambda(3j+1)}(u_0 \oplus x_1), g^0_{\lambda(3j+2)}(v_0 \oplus x_2)\right), \\
&= \left(g^0_{\lambda(3j)}(x_0 \oplus z), g^0_{\lambda(3j+1)}(u_0 \oplus x_1), g^0_{\lambda(3j+2)}(v_0 \oplus x_2)\right), \\
&= \hat{G}_0(j, x \oplus v_z),
\end{aligned}$$

where $v_z = (z, 0, 0) \in \mathbb{Z}_2^{384}$. Similarly, $\hat{G}_z(j,x) = \hat{G}_0(j, x \oplus v_z)$ for $j = \lfloor \frac{tcost}{3} \rfloor - 1$.

Therefore, $\hat{G}_{z,j}(x) = \hat{G}_{0,j}(x \oplus v_z)$ for all $z \in \mathbb{Z}_{t_1}$, $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 1$ and $x \in \mathbb{Z}_2^{384}$. Similar analysis leads to $\tilde{G}_z(j,x) = \tilde{G}_0(j, x \oplus u_z)$ for all $0 \leq j \leq \lfloor \frac{tcost}{3} \rfloor - 1$ where $x, u_z = (0, 0, z) \in \mathbb{Z}_2^{384}$ and $z \in \mathbb{Z}_{t_1}$.

Thus by [9], for all $z \neq z'$,

$$Pr\left[\hat{G}_z(j,x) = \hat{G}_{z'}(j,x)\right] = Pr\left[\tilde{G}_z(j,x) = \tilde{G}_{z'}(j,x)\right] = 0,$$

hence, the adversary can only repeat any sequence with some probability i.e. by targeting a collision at a pair (z, j) and (z', j') such that $z \neq z'$ and $j \neq j'$. Therefore, by Claim 4.1 any successive repeated values will occur with probability 2^{-384} regardless of the input.

Therefore, even very short repeated sequences are unlikely using this method. $\square$

However, this is merely an instance of the more general problem of exploiting repeating sequences in keyed functions with a non-repeating key stream - the adversary has no *apriori* knowledge or control of collisions forcing re-computation of repeating values. In short, if the key stream does not repeat, the adversary must re-compute every successive element of the sequence (including those it observes to be repeating) which eliminates any realistic complexity gain.

Clearly, by the arguments of Claim 4.10, one can view $\bigcirc_{z=0}^{t_1-1}\left(\bigcirc_{j=0}^{\lfloor\frac{tcost}{3}\rfloor-1} G_{z,j}^{X}\right)$ as a composition of $t_1 \times \lfloor\frac{tcost}{3}\rfloor$ distinct permutations. Therefore, non-trivial adversarial complexity gain through exploiting cycles in F is unlikely for fixed $i \in I$ and $p \in P_A$.

4.3 Properties of F

Lemma 4.1. *For any fixed $i \in I$, $f_i \circ S_{2,i} : P_A \to \mathbb{Z}_2^{512}$ is a strong one-way function, where $S_{2,i}(p) = S_2(i,p) \in \mathbb{Z}_2^{384}$ for all $p \in P_A$.*

Proof. Let $\mathcal{E}_{20}^{i} : P_A \times \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ for fixed $i \in I$ be the permutation (for fixed $p \in P_A$) on $\mathbb{Z}_2^{128}$ defined by

$$\mathcal{E}_{20}^{i}(p,x) = \mathcal{E}_{20} \circ \left(\varphi_{20}^{3} \circ S_{2,i} \circ P_1^2, P_2^2\right)(p,x)$$

where P_j^2 is the projective map on 2 elements, $j \in \{1,2\}$, $(p,x) \in P_A \times \mathbb{Z}_2^{128}$ and $\mathcal{E}_{20}$ is the AES-like permutation defined in Section 2.2.

Clearly, for fixed $i \in I$, one can model the success probability for single key attacks against $\mathcal{E}_{20}^{i}$ on those from the AES algorithm over 20 rounds. For the key schedule specific attacks, we have the following.

First, we claim that (for fixed i) pairs of equivalent passwords in $\mathcal{E}_{20}^{i}$ are unlikely. In short, since φ_{Nr}^{3} admits no equivalent keys for all $Nr \geq 9$ by Claim 2.3, we need to show that for all $x \neq y \in P_A$ there exists a negligible function ν such that

$$Pr[S_2(i,x) = S_2(i,y)] \leq \nu(log_2|P_A|).$$

By Claim 2.1, Claim 2.2, Claim 4.1, and Claim 4.8, we have that

$$\begin{aligned} Pr[S_2(i,x) = S_2(i,y)] &= Pr[\mathcal{V}(i,x) = \mathcal{V}(i,y)] \\ &= 2^{-384}, \end{aligned}$$

where for any $w \in P_A$,

$$\begin{aligned} \mathcal{V}(i,w) &= \hat{\mathcal{E}}^{3}_{20,\varphi_{20}^{3}(key_w)} \circ \left(\bigcirc_{z=0}^{t_1-1}\left(\bigcirc_{j=0}^{\lfloor\frac{tcost}{3}\rfloor-1} G_{z,j}^{X^w}\right)\right) \circ \mathcal{U}(v), \\ \mathcal{U} &= \tilde{\mathcal{E}}^{3}_{20,\varphi_{20}^{3}(X_0^w,X_1^w,X_2^w)} \circ \hat{\mathcal{E}}^{3}_{7,(X_{t_0-8}^w,X_{t_0-7}^w,\cdots,X_{t_0-1}^w)}, \end{aligned}$$

X^w is the X-array for w, $v = (0,0,0) \in \mathbb{Z}_2^{384}$ and $key_w = w||0^{384-|w|}$. Therefore, pairs of equivalent passwords are unlikely.

Second, we claim that for fixed i, related-password attacks are unlikely. This follows from Claim 2.4 and Claim 4.7. Finally, we claim that for fixed i, the Biclique attack on $\mathcal{E}_{20}^i$ is unlikely. This follows from the high diffusion and nonlinearity in φ_{20}^3 plus some element of one-wayness in φ_{20}^3 for the necessary *few rounds* over which propagation of detectable differences is likely [4].

Moreover, the use of round constants ensures that the symmetry of the round function is eliminated. Therefore, high probability key schedule attacks against $\mathcal{E}_{20}^i$ for fixed i are unlikely.

For brevity, let $g = f_i \circ S_{2,i}$ and $i \in I$ be fixed. Then, for any $x \neq y \in P_A$,

$$\begin{aligned} Pr[g(x)=g(y)] = \quad & Pr[g(x)=g(y)|S_2(i,x)=S_2(i,y)] \cdot Pr[S_2(i,x)=S_2(i,y)] \\ & + Pr[g(x)=g(y)|S_2(i,x)\neq S_2(i,y)] \cdot Pr[S_2(i,x)\neq S_2(i,y)] \end{aligned}$$

whence,

$$\begin{aligned} Pr\left[f_i \circ S_{2,i}(x) = f_i \circ S_{2,i}(y)\right] &= 2^{-384} + 2^{-512}\cdot(1-2^{-384}) \\ &\leq 2^{-384}(1+2^{-128}). \end{aligned}$$

Therefore, for any random $y \in P_A$, we must have that (on average)

$$\frac{|\{x \in P_A : x \neq y, f_i \circ S_{2,i}(x) = f_i \circ S_{2,i}(y)\}|}{|P_A|-1} \leq 2^{-384}(1+2^{-128}).$$

Hence,

$$|\{x \in P_A : x \neq y, f_i \circ S_{2,i}(x) = f_i \circ S_{2,i}(y)\}| \leq (|P_A|-1)\cdot\left(2^{-384}(1+2^{-128})\right).$$

Therefore, we claim that for fixed i

$$\begin{aligned} Adv_{\mathcal{E}_{20}^i}^{prf}(A_{t',4}) &\leq c_1 \frac{t^{'}/T_{f_i\circ S_{2,i}} \cdot \left(((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\right)}{|P_A|} + \frac{12}{2^{129}} \\ &\leq c_1 \frac{t^{'}/T_{f_i\circ S_{2,i}} \cdot \left((1-\frac{1}{|P_A|})\cdot(1+2^{-128})+1\right)}{2^{384}} + \frac{12}{2^{129}} \end{aligned}$$

for any PRF adversary A that makes at most 4 oracle queries and runs in time at most $t^{'} = O\left(\log_2(|P_A|)+128+128+T_{f_i\circ S_{2,i}}\right)$, where $T_{f_i\circ S_{2,i}}$ is the time for a single execution of $f_i \circ S_{2,i}$ (about 4 encryption runs of $\mathcal{E}_{20}$ plus one run of the key schedule, $\varphi_{20}^3 \circ S_{2,i}$) and $c_1 \in \{0,1\}$ [9].

On the other hand, for any inverter h for $f_i \circ S_{2,i}$, define

$$Adv_{f_i\circ S_{2,i},h}^{owf}(t) = Pr\left[f_i \circ S_2(i,k^{'}) = y;\ k \xleftarrow{\$} P_A;\ y = f_i \circ S_2(i,k);\ k^{'} = h(y)\right]$$

where h runs in time at most t [9].

Clearly, for any inverter h of $f_i \circ S_{2,i}$, we can construct a prf-adversary A for $\mathcal{E}_{20}^i$ as follows.

Adversary A^f
  Compute $y = (f(C_0), f(C_1), f(C_2), f(C_3))$
  Run h to obtain $k^{'} = h(y)$
  If $f_i \circ S_{2,i}(k^{'}) = y$ then
    Return 1
  else
    Return 0

Since A has oracle access to the function instance f of either $\mathcal{E}_{20}^i$ or $\text{Rand}^{128\to 128}$ it can compute $y = f(x)$ for any $x \in \mathbb{Z}_2^{128}$. Therefore, it can run h as a subroutine which recovers the key with probability $Adv_{f_i \circ S_{2,i},h}^{owf}(t)$ whenever f is an instance of $\mathcal{E}_{20}^i$ and where t is the maximum running time for h.

Moreover, since $f_i \circ S_{2,i}$ is a public function, A can compute $f_i \circ S_{2,i}(k^{'})$ to confirm the result [1].

Therefore,

$$\begin{aligned} Pr[f \xleftarrow{\$} \mathcal{E}_{20}^i : A^f = 1] &= Adv_{f_i \circ S_{2,i},h}^{owf}(t) \\ Pr[f \xleftarrow{\$} \text{Rand}^{128\to 128} : A^f = 1] &= \frac{1}{2^{512}} \end{aligned}$$

Thus, we must have,

$$\begin{aligned} Adv_{\mathcal{E}_{20}^i}^{prf}(A) &= Pr[f \xleftarrow{\$} \mathcal{E}_{20}^i : A^f = 1] - Pr[f \xleftarrow{\$} \text{Rand}^{128\to 128} : A^f = 1] \\ &= Adv_{f_i \circ S_{2,i},h}^{owf}(t) - \frac{1}{2^{512}}. \end{aligned}$$

Therefore,

$$\begin{aligned} Adv_{\mathcal{E}_{20}^i}^{prf}(4, t^{'}) + \frac{1}{2^{512}} &\geq \max_h \{Adv_{f_i \circ S_{2,i},h}^{owf}(t)\} \\ &= Adv_{f_i \circ S_{2,i}}^{owf}(t). \end{aligned}$$

Hence, by Proposition 2.5 of [1],

$$
\begin{aligned}
Adv^{owf}_{f_i \circ S_{2,i}}(t) &\le \frac{t'/T_{f_i \circ S_{2,i}} \cdot \big((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\big)}{|P_A|} + \frac{12}{2^{129}} + \frac{1}{2^{512}} \\
&\le \frac{t'/T_{f_i \circ S_{2,i}} \cdot \big((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\big)}{|P_A|} + \frac{13}{2^{129}} \\
&= \frac{t/T_{f_i \circ S_{2,i}} \cdot \big((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\big)}{|P_A|} + \frac{13}{2^{129}} \\
&\quad + \frac{Q/T_{f_i \circ S_{2,i}} \cdot \big((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\big)}{|P_A|} \\
&\le \frac{t/T_{f_i \circ S_{2,i}} \cdot \big((|P_A|-1)\cdot 2^{-384}(1+2^{-128})+1\big)}{|P_A|} + \varepsilon
\end{aligned}
$$

where $t' = t + Q$, $Q = O\big(\log_2(|P_A|) + 128 + 128 + T_{f_i \circ S_{2,i}}\big)$ and ε is some fixed constant [1, 9].

Therefore, $f_i \circ S_{2,i}$ is a strong one-way function i.e. for $\log_2 |P_A|$ large enough, $Adv^{owf}_{f_i \circ S_{2,i}}(t)$ is a negligible function of t and $\log_2 |P_A|$ (ε, for example, becomes a *small constant*). $\square$

In short, we are claiming that no PRF adversary for $\mathcal{E}^i_{20}$ using 4 plaintexts apart from the birthday paradox exists i.e. according to current literature on attacks against the AES [4, 5].

Proposition 4.2. *The collection of functions $F = \{f_i : \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{512}\}_{i \in I}$, with S_1 and S_2 as defined above is a collection of strong one-way functions under non-parameter modifying adversaries.*

Proof. Follows from Lemma 4.1. $\square$

Proposition 4.3. *The collection of functions $F = \{f_i : \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{512}\}_{i \in I}$, with S_1 and S_2 as defined above is a collection of strong one-way functions under parameter modifying adversaries.*

Proof. First, we consider related-salt attacks.

Clearly, by Claim 2.4 such attacks are unlikely since the attacker is unlikely to transition non-trivial salt differences through φ^2_{20}. Further, by Claim 4.8 and the differential propagation ratio of $\mathcal{E}_{4,\vartheta_4(0)}$ a related-salt attack against F is unlikely.

For related-T attacks, it is clear that

$$\bigcirc_{z=0}^{t_1-1}\left(\bigcirc_{j=0}^{\lfloor \frac{tcost}{3} \rfloor - 1} G^X_{z,j}\right) \neq \bigcirc_{z=0}^{t_1'-1}\left(\bigcirc_{j=0}^{\lfloor \frac{tcost'}{3} \rfloor - 1} G^{X'}_{z,j}\right)$$

for any distinct pairs of time parameters, (t_0, t_1) and $(t_0^{'}, t_1^{'})$. Therefore, in general, related-T attacks present two inequivalent permutations to the adversary which by Claim 4.1 and Claim 4.10 ensures the results are only related by some probability function - since the output of S_2 does not interact directly with the plaintexts and, thus, with the output of F. However, we need to assess the adversarial complexity gain through computation of fixed password and salt values under distinct time parameters [27].

Clearly, assuming fixed salt and password values, each $(t_0, t_1) \in T$ changes each of the first four entries of the X-array. Therefore, by arguments of Claim 4.7, every set of four consecutive entries of the X-array for any two distinct T values will be distinct which ensures all the $G_{z,j}$ values are component wise distinct between the pairs. On the other hand, $\mathcal{E}_{4,\vartheta_4(0)}$ ensures that such differences have very high propagation ratio over the entire $t_0 \geq 2^{20}$ elements of the X-array.

Therefore, by the arguments in and after Claim 4.10, the adversary needs to re-compute the final two *for loops* of S_2 for each element in the pair.

Combined attacks encounter similar problems as the adversary needs to successfully mount a high probability related key attack against $\mathcal{E}_{20}$ with salt differences as key differences and T XOR differences as plaintext differences before finally navigating through the changing last two *for loops* environment. However, this is unlikely by Claim 2.4.

Therefore, related-I attacks are unlikely. □

Lemma 4.4. *The M3dcrypt password hashing function F achieves near ideal security for any non-uniform password distribution D.*

Proof. We need to show that F is a (t, ϵ)-secure password hashing function where ϵ is as close to $\frac{t}{|P_A|}$ as possible under the uniform distribution (see Theorem 1.2 in Section 1.2).

By Lemma 4.1, F is a (t, ϵ)-secure password hashing function under the uniform distribution, where

$$\begin{aligned}\epsilon &= \frac{t \cdot \big((|P_A| - 1) \cdot 2^{-384}(1 + 2^{-128}) + 1\big)}{|P_A|} \\ &\leq \frac{t \cdot (2 + 10^{-38})}{|P_A|}.\end{aligned}$$

since $|P_A| \leq 2^{384}$ and only key recovery PRF adversaries are admissible. Therefore, F achieves near ideal security for any non-uniform password distribution D [26]. □

5 Implementation Issues

5.1 Software Implementation

The M3dcrypt password scheme is designed to exploit the high efficiency Advanced Encryption Standard New Instructions (AES-NI) through a design that makes extensive use of the AES encryption round function (AESENC).

Therefore, M3dcrypt admits efficient implementation on all platforms including those with modern features such as Single Instruction Multiple Data (SIMD) and multicore CPUs [6, 2].

For completion, an example non-AES-NI implementation on a 1.6 GHZ Intel Core 2 Duo Processor running the GCC compiler completes 4.742 evaluations of M3dcrypt per second (using minimum parameters i.e. $t = (2^{20}, 1) \in T$). In comparison, at creation in 1977 [21], *crypt* could be evaluated about 3.6 times per second on a VAX-11/780.

5.2 Hardware Implementation

The availability of large random access memory (RAM) on general purpose microprocessors shifts the implementation bottleneck from random access memory (RAM) to optimal implementation of the cryptographic primitive.

On the contrary, we can assume that efficient hardware for primitives in wide spread use exist (e.g. standardised algorithms such as the AES). Possibilities for further optimisation (e.g. external pipelining and/or other extensive parallelism) are contingent on the availability and cost of RAM.

However, by Claims 4.2, 4.3 and 4.6, the high entropy X array ensures that extensive time/memory trade-offs increase the number of auxiliary computations required to process further X_k values, $0 \leq k \leq t_0 - 1$, in the computation of $v = (v_0, v_1, v_2) \in \mathbb{Z}_2^{384}$.

Therefore, assuming large memory requirement for X, massively parallel key search machines may be [area-time] costly [20, 13].

6 Dedication

To one in whom all things are at once both meaningful and meaningless, all labours both futile and glorious; and to another of whom I presumed to know much, yet perceived little until the pulling down of this tent.

7 Conclusion

We have described a new password hashing function which is secure as long as $\mathcal{E}_{20} \circ \left(\varphi_{20}^3 \circ S_{2,i} \circ P_1^2, P_2^2\right) : P_A \times \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ is a secure PRF for all adversaries using at most 4 oracle queries. Furthermore, we have shown that F is close to ideal security for any password distribution D.

References

[1] M. Bellare, J. Killian and P. Rogaway, *The Security of the Cipher Block Chaining Message Authentication Code*, Journal of Computer and System Sciences, Vol. 61 No. 3, pp. 362-399, 2000.

[2] R. Benadjila, *Use of the AES Instruction Set*, ECRYPT II AES Day, October 2012.

[3] E. Biham and A. Shamir, *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, 4, 1, pp. 372, 1991.

[4] A. Bogdanov, D. Khovratovich and C. Rechberger, *Biclique Cryptanalysis of the Full AES*, ASIACRYPT 2011, LNCS 7073, pp. 344-371, 2011.

[5] J. Daemen and V. Rijmen, *On the Related-key Attacks Against AES*, Proceedings of the Romanian Academy, Series A, Volume 13, Number 4/2012, pp. 395400, 2012.

[6] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES Submission, http://www.nist.org/aes, 1999.

[7] C. Davis and R. Ganesan, *BApasswd: A New Proactive Password Checker*, 16^{th} National Computer Security Conference, September 1993.

[8] J. Gallier and A. Hicks, *The Theory of Languages and Computation*, Lecture Notes, University of Pennsylvania, http://www.cis.upenn.edu/ jean/gbooks/toc.pdf, Accessed December 2014.

[9] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*, July 2008.

[10] T. Jiang et al, *Formal Grammars and Languages*, http://www.cs.ucr.edu/ jiang/cs215/tao-new.pdf, Accessed December 2014.

[11] B. Kaliski, *PKCS #5: Password-Based Cryptography Specication Version 2.0*, RFC 2898, 2000.

[12] B. Kaliski et al, *Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)*, International Association for Cryptologic Research, Journal of Cryptology, Vol. 1, pp. 3-36, 1988.

[13] J. Kelsey, B. Schneier, C. Hall and D. Wagner, *Secure Applications of Low-Entropy Keys*, Proceedings of the First International Workshop ISW 97, Springer-Verlag, 1998.

[14] J. Kelsey, B. Schneier and D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Lecture Notes in Computer Science, 1109, pp. 237-251, 1996.

[15] D. Klein. *Foiling the Cracker: A Survey of and Improvements to Password Security*, Proceedings, UNIX Security Workshop II, August 1990.

[16] H. Krawczyk, *Cryptographic Extraction and Key Derivation: The HKDF Scheme*, Crypto'2010, LNCS 6223, 2010.

[17] R. Morris and K. Thomson, *Password Security: A Case History*, Communications of the ACM, Vol. 22 No. 11, November 1979.

[18] NIST, *FIPS-197: Advanced Encryption Standard*, National Institute of Standards and Technology (NIST), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001.

[19] S. Park, S. H. Sang, S. Lee, and J. Lim, *Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES*, Fast Software Encryption 2003, LNCS 2887, pp. 247-260, Springer-Verlag, 2003.

[20] C. Percival and S. Josefsson, *The Scrypt Password-Based Key Derivation Function*, IETF Internet Draft, 2012.

[21] N. Provos and D. Mazieres. *A Future-Adaptable Password Scheme*, USENIX Annual Technical Conference, USENIX 99, The Advanced Computing Systems Association, 1999.

[22] V. Rijmen and P. Barreto. *The Anubis Block Cipher*, Submission to the NESSIE Project, March 2000.

[23] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Second Edition, 1996.

[24] W. Stallings. *Cryptography and Network Security: Principles and Practice*, Prentice Hall, Second Edition, 1998.

[25] D. R. Stinson, *Cryptography: Theory and Practice*, Second Edition, Chapman & Hall, 2002.

[26] D. Wagner and I. Goldberg, *Proofs of Security For The UNIX Password Hashing Algorithm*, In Advances in Cryptology - Asiacrypt 00, Springer-Verlag, 2000.

[27] F.F. Yao and Y.L. Yin, *Design and Analysis of Password-Based Key Derivation Functions*, CT-RSA 2005.