# The M3dcrypt Password Scheme

Isaiah Makwakwa
imakwakwa@gmail.com

**Abstract**

M3dcrypt is a password authentication scheme built around the advanced Ecryption Standard (AES) and the arcfour pseudorandom function. It uses up to 256-bit pseudorandom salt values and supports 48-byte passwords.

## 1 Introduction

The user induced probability distribution, $\mathcal{D}_{user}$, on the password space, $\mathcal{K}_p$ (e.g a subset of strings from the 95 printable 7-bit ASCII characters) has inherent low entropy [10, 17, 16]. Therefore, $\mathcal{K}_p$ as a source of cryptographic key material is vulnerable to both *brute force* and *dictionary* attacks [9, 20, 7, 14].

**Definition 1.1.** *A password scheme, $\{PS_s\}_{s \in S}$, is a function family such that for any salt value $s \in S$ there exists a one-way function,*

$$PS_s : \mathcal{K}_p \to \{0, 1\}^L.$$

A *function family* is necessary, for since $\mathcal{D}_{user}$ has low entropy, a deterministic function as opposed to a pseudorandom function family, allows precomputation of tables of password hashes for all or part of the password space. Thus, it facilitates on the fly computation of pre-images.

However, high functional dependency on auxiliary randomness (often non-secret), $s \in S$, from a large space increases the uncertainty associated with the scheme. Indeed, with a large salt space, the space-time complexity for complete or partial precomputation might be out of reach of even the most well resourced of adversaries [7, 9, 14, 16].

For *one-way function*, we require that within feasible computational effort, the adversary's inversion probability (i.e. the probability of a successful password recovery given a password hash) will remain below a certain [small] threshold [6, 19].

However, in practice, the above postulation requires some qualification. Consider a password scheme based on a cryptographic hash of [a concatenation of] the user password and some function of the salt value. Clearly, under the random oracle model, such a scheme can within certain computational parameters be considered secure. However, assuming adversarial knowledge of $s \in S$ and the password hash, the time complexity for exhaustive and/or dictionary search may be too low for certain $\mathcal{D}_{user}$. Indeed, by Moore's law, the adversarial distinguishing probability doubles every 18 months [6, 16, 14].

On the contrary, certain elements of *key stretching* such as the iterative application of some cryptographic primitive(s) allows the adversarial distinguishing probability to remain constant even with increasing computational power [9]. In particular, it is known that key stretching techniques, in the absence of design flaws such as narrow pipes and reusable internal values, allow for a quantifiable increase in the complexity of dictionary and brute force attacks [9, 20].

The above not withstanding, a moderately resourced adversary can build special purpose key search machines (e.g. the Electronic Frontier Foundation's DES cracker [14] and M. Wiener's design for a DES cracker [16]) that dramatically reduce the [area-time] cost of brute force attacks.

Moreover, by Moore's law [16], [cryptographic] circuits not only become faster but cheaper and smaller allowing for greater parallelism and dramatic growth in the economies of scale available to the attacker. Therefore, hardware-frustrating techniques such as memory and/or expensive operations may be necessary for imposing cost constraints on custom circuits while ensuring efficiency of computation on general purpose processors [9, 13].

In this paper, a new password based key derivation 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 scheme, Section 4 analyses the security of the scheme and Section 5 explores some implementation issues.

### 1.1 The M3dcrypt Byte Ordering and Notation

The M3dcrypt password scheme assumes little endian byte ordering. However, big endian byte ordering can also be used so long consistency is ensured for all functions and constants [15].

Further, the M3dcrypt password scheme adopts the notation $V_i$ for the $i^{th}$ element of any array $V$.

## 2 Preliminaries

The M3dcrypt password scheme is based on the Advanced Encryption Standard (AES) algorithm [11]. In particular, M3dcrypt implements a set of AES-like permutations

$$\mathcal{E}_{(Nr,Y)} : \mathbb{Z}_2^{128} \rightarrow \mathbb{Z}_2^{128}$$

where $Nr$ denotes the number of rounds of the AES encryption function and $Y$ is an array of $Nr + 1$ 128-bit round subkeys.

In particular,

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

where $\sigma_k(state) = AddRoundKey(state, Y_k)$, $\gamma(state) = ByteSub(state)$, $\tau(state) = ShiftRow(state)$ and $\theta(state) = MixColumn(state)$ [5, 18].

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. □

## 2.1 The M3dcrypt Auxilliary Key Schedule

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

$$S = \theta \circ \tau \circ \gamma$$

denote the unkeyed AES round function.

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

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

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

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

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

Then, the auxilliary key schedule *key initialisation function*

$$I^f : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128 \times 2m},$$

is defined by the following algorithm.

**Algorithm 1: Key Initialisation Function, $I^f$**

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

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

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

The auxilliary key schedule *key extraction function*

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

is defined by the following algorithm.

**Algorithm 2: Key Extraction Function, $f^X$**

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

$f^X(k_0, k_1, \cdots, k_{m-1})$**:**

$p := 0$; $\omega_{p-1} := 0$; $\phi_{p-1} := 0$

**while** $(p < (Nr - 2m + 1))$ **do**

$\omega_p := S\left(\omega_{p-1} \oplus (p+1) \oplus \left(\bigoplus_{i=p}^{p+m-1} k_i\right)\right)$

$k_p := S(\phi_{p-1} \oplus \omega_p)$

$\phi_p := S(\phi_{p-1} \oplus k_p)$

$p := p + 1$

**end while**

**Return** $(k_0, k_1, k_2, \cdots, k_{Nr-2m})$

Finally, define $\varphi_{Nr} : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128\times(Nr+1)}$ the $Nr$-round ($Nr \geq 3m$) auxilliary key schedule for $128m$-bit master keys by

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

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

## 2.2 The M3dcrypt Constants

The M3dcrypt constants are based on the first four subkeys of the AES-128 key schedule for master key 0, $C = \vartheta_3(0)$. Therefore,

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

## 2.3 Properties of the Auxilliary Key Schedule

**Claim 2.3.** *For $Nr \geq 3m$, pairs of equivalent keys in $\varphi_{Nr}$ 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 $\varphi_{Nr}(key^0) = \varphi_{Nr}(key^1)$.

Since $\pi^m(key^0) \neq \pi^m(key^1)$ are part of the subkey sequence(s), $\varphi_{Nr}(key^0) \neq \varphi_{Nr}(key^1)$ for all $key^0 \neq key^1 \in \mathbb{Z}_2^{128m}$ and $Nr \geq 3m$. □

**Claim 2.4.** *For $Nr \geq 3m$ and $m < 15$, related-key differential attacks in $\varphi_{Nr}$ are unlikely.*

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

Since, on average, a brute force attack requires $2^{n-1}$ rekeyings [17, 15], any $n$-bit key schedule in which transitioning non-trivial differences has maximum probability $2^{1-n}$ is resilient against the attack.

However, in effect, this merely re-states the requirement for key schedule resilience against differential attacks [8, 3].

On the other hand, bearing in mind the arguments of [12, 2], we note that $\pi^m$ has differential propagation ratio at most $2^{-120(m+1)}$. Hence, resistance against related-key attacks holds whenever the following inequality holds

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

and thus whenever $m < 15$. $\square$

# 3 The M3dcrypt Password Hashing Algorithm

Let $salt \in \mathbb{Z}_2^k$, $128 \leq k \leq 256$, be the *arcfour* generated salt value zero padded to 256 bits if necessary, *passwd* be the user password, $2^{20} \leq m\_cost \leq 2^{31}$ (a power of two) be the configurable memory parameter, $2^3 \leq tfactor \leq 2^5$ be the configurable time factor and $X$ be an array of *m_cost* 128-bit values defined below.

Then the M3dcrypt password hashing function is the AES based variant of the *bcrypt* design [14] defined below.

## 3.1 The M3dcrypt Key Schedule Parameters

Let $2^{20} \leq m\_cost \leq 2^{31}$ and $2^3 \leq tfactor \leq 2^5$ be the configurable memory and time parameters respectively . Further, let

$$t\_cost = \frac{m\_cost}{tfactor},\ lt\_cost = \log_2(t\_cost),\ rt0 = 1, \text{ and } rt1 = 3$$

be fixed, then the rest of the M3dcrypt key schedule parameters depend on the value of *lt_cost* as follows.

| lt_cost | skey | rt2 | rt3 |
|---|---|---|---|
| $8-16$ | $0xD09788FD$ | 6 | 2 |
| $17-28$ | $0xD09788FD$ | 8 | 15 |
| 29 | $0x27E6FB94$ | 8 | 15 |
| 30 | $0x2C8DB305$ | 8 | 15 |
| 31 | $0xD09788FD$ | 8 | 16 |

### 3.2 The M3dcrypt Key Schedule

The M3dcrypt key schedule algorithm $\mathcal{V} : \mathbb{Z}_2^{384} \to \mathbb{Z}_2^{128\times 21}$ follows the Anubis design based on a main *key selection function* complemented by a *key evolution function* [15].

For any fixed integer $z \in \mathbb{Z}$, let $\psi_z : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ be defined by

$$\psi_z(x) = x \oplus z,$$

where $z$ is considered as a 128-bit little endian integer.

Then the M3dcrypt *key evolution function*,

$$\chi : \mathbb{Z}_2^{384} \times \mathbb{Z}_2^{256} \times \mathbb{Z} \to \left(\mathbb{Z}_2^{128}\right)^{m\_cost},$$

is defined by Algorithm 3 below.

**Algorithm 3: Key Evolution Function, $\chi$**

**Require:** $key := (passwd||0^{384-|passwd|}) \in \mathbb{Z}_2^{384}$, $salt \in \mathbb{Z}_2^{256}$,
$2^{20} \leq m\_cost \leq 2^{31}$.

$\chi(passwd, salt, m\_cost)$**:**
**for** $z = 0$ **to** 3 **do**
$X_z := \mathcal{E}_{(4,\vartheta_4(0))} \circ \psi_z \circ \mathcal{E}_{(20,\varphi_{20}(key))} \circ \mathcal{E}_{(16,\varphi_{16}(salt))}(C_z)$.
**end for**
**for** $z = 4$ **to** $m\_cost - 1$ **do**
$X_z := \mathcal{E}_{(4,\vartheta_4(0))} \circ \psi_z(X_{z-1} \oplus X_{z-4})$.
**end for**

**Return** $X$

For the *key selection function* we require the following.

Let $ROT(x, k)$ denote the *right cyclic shift* of $x \in \mathbb{Z}_2^{t\_cost}$ by $k$ bits and $\lambda : \mathbb{Z}_{t\_cost} \to \mathbb{Z}_{m\_cost}$ denote the the linear injection defined by

$$\lambda(x) \quad = \quad \varepsilon \circ l(x),$$

where

$$l(x) = ROT(x, rt1) \oplus skey$$

and

$$\varepsilon(x) \quad = \quad tfactor \times (ROT(x, rt0) \oplus ROT(x, rt2) \oplus ROT(x, rt3))$$

for all $x \in \mathbb{Z}_{t\_cost}$.

Define $f_p^*, g_p^* : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$, the AES-like permutations defined by

$$\begin{aligned} g_p^* &= \mathcal{E}_{\left(tfactor-1,(X_{\lambda(p)},X_{\lambda(p)+1},X_{\lambda(p)+2},\cdots,X_{\lambda(p)+tfactor-1})\right)}, \\ f_p^* &= \mathcal{E}_{\left(tfactor-1,(X_{\lambda(p)},X_{\lambda(p)+1},X_{\lambda(p)+2},\cdots,X_{\lambda(p)+tfactor-1})\right)} \end{aligned}$$

where $0 \leq p \leq t\_cost - 1$.

Let $\hat{g}_p^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ denote the domain extension defined by

$$\hat{g}_p^m(x) = (g_p(x), g_{p+1}(x), g_{p+2}(x), \cdots, g_{p+m-1}(x))$$

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

$$g_{p+k}(x) = g_{p+k}^*(g_{p+k-1}(x) \oplus x_k)$$

$0 \leq k \leq m-1$ is recursively defined by setting $g_{p-1}(z) = 0$, for all $z \in \mathbb{Z}_2^{128m}$ and all values of $p$.

Similarly, let $\tilde{f}_p^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ denote the domain extension defined by

$$\tilde{f}_p^m(x) = (f_p(x), f_{p+1}(x), f_{p+2}(x), \cdots, f_{p+m-1}(x))$$

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

$$f_{p+k}(x) = f_{p+k}^*(f_{p+k+1}(x) \oplus x_k)$$

$0 \leq k \leq m-1$ is recursively defined by setting $f_{p+m}(z) = 0$, for all $z \in \mathbb{Z}_2^{128m}$ and all values of $p$.

Clearly, the above definitions require that $\tilde{f}_p^m$ and $\hat{g}_p^m$ have *instance dependence* i.e. given consecutive computations $\hat{g}_p^m(x)$ and $\hat{g}_{p+m}^m(x)$,

$$g_{p+m-1}(x) = \begin{cases} g_{p+m-1}^*(g_{p+m-2}(x) \oplus x_{m-1}) & \text{in } \hat{g}_p^m(x) \\ 0 & \text{in } \hat{g}_{p+m}^m(x). \end{cases}$$

for any fixed input $x \in \mathbb{Z}_2^{128m}$.

**Claim 3.1.** *The domain extensions $\tilde{f}_p^m, \hat{g}_p^m : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ are permutations.*

*Proof.* Similar to Claim 2.1. □

Moreover, since for any $1 \leq m' < m$ we can unambiguously (by some appropriate isomorphism) express $\mathbb{Z}_2^{128m}$ as $\mathbb{Z}_2^{128m'} \times \mathbb{Z}_2^{128(m-m')}$, we obtain trivial extensions $\tilde{f}_p^{\prime m'} : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$, defined by

$$\tilde{f}_p^{\prime m'}(x, y) = \left(\tilde{f}_p^{m'}(x), y\right)$$

where $x \in \mathbb{Z}_2^{128m'}$ and $y \in \mathbb{Z}_2^{128(m-m')}$. The case of $\hat{g}_p'^{m'}$ is similar.

Let $\Psi^m_{(X,t\_cost,tfactor)} : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ denote the permutation defined by

$$\Psi^m_{(X,t\_cost,tfactor)} = \tilde{f}'^{r-m'}_{2m\times\lfloor\frac{t\_cost}{2m}\rfloor+m} \circ \hat{g}'^{m'}_{2m\times\lfloor\frac{t\_cost}{2m}\rfloor} \bigcirc_{i=0,p=2mi}^{\lfloor\frac{t\_cost}{2m}\rfloor-1} \left(\tilde{f}^m_{p+m} \circ \hat{g}^m_p\right)$$

where $r = (t\_cost \,\%\, 2m)$ and $m' = Minimum(m,r)$ with the convention $\hat{g}_s'^0(z) = \tilde{f}_k'^0(z) = z$ for all $s,k \geq 0$ and $\forall z \in \mathbb{Z}_2^{128m}$.

For the special case of the M3dcrypt password hashing algorithm, $\Psi^3_{(X,t\_cost,tfactor)}$ can be algorithmically defined by Algorithm 4 below.

**Algorithm 4: Algorithmic View of $\Psi^3_{(X,t\_cost,tfactor)}$**

**Require:** $v = (v_0, v_1, v_2) \in \mathbb{Z}_2^{384}$, $2^{17} \leq t\_cost \leq 2^{28}$, $X \in \left(\mathbb{Z}_2^{128}\right)^{m\_cost}$
$2^3 \leq tfactor \leq 2^5$.

$\Psi^3_{(X,t\_cost,tfactor)}(v)$**:**
- $ts := 0; p := 0; z := 0$
- **while** $z < t\_cost$ **do**
  - $p := \lambda(z)$
  - $key := (X_p, X_{p+1}, X_{p+2}, \cdots, X_{p+tfactor-1})$
  - $ts := z \,\%\, 6$
  - **if** $ts < 3$ **then**
    - **if** $ts = 0$ **then**
      - $v_0 := \mathcal{E}_{(tfactor-1,key)}(v_0)$
    - **else**
      - $v_{ts} := \mathcal{E}_{(tfactor-1,key)}(v_{ts} \oplus v_{ts-1})$
    - **end if**
  - **else**
    - **if** $ts = 3$ **then**
      - $v_2 := \mathcal{E}_{(tfactor-1,key)}(v_2)$
    - **else**
      - $v_{5-ts} := \mathcal{E}_{(tfactor-1,key)}(v_{5-ts} \oplus v_{6-ts})$
    - **end if**
  - **end if**
  - $z := z + 1$
- **end while**

**Return** $v$

Further, let $key = (X_{(m\_cost-tfactor)}, X_{(m\_cost-tfactor+1)}, \cdots, X_{(m\_cost-1)})$ and set $g = \mathcal{E}_{(tfactor-1,key)}$. Define

$$\hat{g}^m_X = \hat{g}^m$$

where $\hat{g}^m$ is defined in Section 3.

Similarly, let $f = \mathcal{E}_{(20,key)}$ where $key = (X_0, X_1, \cdots, X_{20})$. Define

$$\tilde{f}^m_X = \tilde{f}^m$$

where $\tilde{f}^m$ is defined in Section 3.

Finally, let $key \in \mathbb{Z}_2^{384}$ and $g = \mathcal{E}_{(20,\varphi_{20}(key))}$, define

$$\hat{g}^m_{key} = \hat{g}^m,$$

where $\hat{g}^m$ is defined in Section 3.

Let $key \in \mathbb{Z}_2^{384}$ and $\Pi^m_{(X,t\_cost,tfactor)} : \mathbb{Z}_2^{128m} \to \mathbb{Z}_2^{128m}$ be the permutation defined by

$$\Pi^m_{(X,t\_cost,tfactor)} = \hat{g}^m_{key} \circ \Psi^m \circ \tilde{f}^m_X \circ \hat{g}^m_X,$$

then the M3dcrypt *key selection function* is the map

$$\varphi_{20} \circ \Pi^3_{(X,t\_cost,tfactor)} : \{0\} \to \mathbb{Z}_2^{128\times 21}$$

where $X \in \left(\mathbb{Z}_2^{128}\right)^{m\_cost}$, $2^3 \leq tfactor \leq 2^5$ and $2^{17} \leq t\_cost \leq 2^{28}$.

Therefore, the M3dcrypt key schedule algorithm

$$\mathcal{V} : \mathbb{Z}_2^{384} \times \mathbb{Z}_2^{256} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}_2^{128\times 21}$$

is defined by Algorithm 5 below.

**Algorithm 5: M3dcrypt Key Schedule Algorithm, $\mathcal{V}$**

**Require:** $passwd \in \mathcal{K}_p$, $salt \in \mathbb{Z}_2^{256}$, $2^{20} \leq m\_cost \leq 2^{31}$, $2^3 \leq tfactor \leq 2^5$

$\mathcal{V}(passwd, salt, m\_cost, tfactor)$**:**
$key := passwd||0^{384-|passwd|}$
$t\_cost := \frac{m\_cost}{tfactor}$
$X := \chi(key, salt, m\_cost)$

**Return** $\varphi_{20} \circ \Pi^3_{(X,t\_cost,tfactor)}(0)$

### 3.3 The M3dcrypt Password Hashing Function

Let $salt \in \mathbb{Z}_2^{256}$, $2^{20} \leq m\_cost \leq 2^{31}$ and $2^3 \leq tfactor \leq 2^5$. Then the M3dcrypt password hashing function,

$$\text{M3dcrypt\_hash}_{(salt,m\_cost,tfactor)} : \mathcal{K}_p \to \mathbb{Z}_2^{512},$$

is defined by Algorithm 5 below.

**Algorithm 5: The M3dcrypt_hash Algorithm**

**Require:** $passwd \in \mathcal{K}_p$, $salt \in \mathbb{Z}_2^{256}$, $2^{20} \leq m\_cost \leq 2^{31}$,
$2^3 \leq tfactor \leq 2^5$, $C$ from Section 2.2.

**M3dcrypt_hash**$_{(salt,m\_cost,tfactor)}(passwd)$**:**
  $g := \mathcal{E}_{(20,\mathcal{V}(passwd,salt,m\_cost,tfactor))}$
  **for** $i := 0$ **to** 3 **do**
    $h_i := g^2(C_i)$
  **end for**

  **Return** $(h_0, h_1, h_2, h_3)$

# 4 Security Analysis

For this section, we require the following properties (Claim 4.1 and Claim 4.2) of random permutations.

**Claim 4.1.** *For any random permutation* $\pi : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ *and any two elements* $x, y \in \mathbb{Z}_2^n$, $Pr[\pi(x) = y] = 2^{-n}$ *[6].*

**Claim 4.2.** *For any two random permutations* $\pi_0, \pi_1 : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ *and any two elements* $x, y \in \mathbb{Z}_2^n$,

$$Pr[\pi_0(x) = \pi_1(y)] = \begin{cases} 2^{-n} & \pi_0 \neq \pi_1 \\ 1 & \pi_0 = \pi_1 \text{ and } x = y \\ 0 & \pi_0 = \pi_1 \text{ and } x \neq y \end{cases}$$

*Proof.* Since the second and last cases are clear, we consider the case $\pi_0 \neq \pi_1$.

We have,

$$\begin{aligned} Pr[\pi_0(x) = \pi_1(y)] &= \sum_{z \in \mathbb{Z}_2^n} Pr[\pi_0(x) = z | \pi_1(y) = z] \cdot Pr[\pi_1(y) = z] \\ &= 2^n \cdot \frac{1}{2^{2n}} \\ &= 2^{-n}. \end{aligned}$$

□

## 4.1 Properties of $\chi$

**Claim 4.3.** *For any fixed random password and salt value, any set of six consecutive elements of the $X$ array has at least two distinct elements.*

*Proof.* We prove by contradiction. Let $X_{z-4} = X_{z-3} = X_{z-2} = \cdots = X_z = X_{z+1}$, $(4 \leq z \leq m\_cost - 2)$ be a set of 6 consecutive elements of the $X$ array for a fixed random password and salt value.

Then we must have

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

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

**Claim 4.4.** *For any fixed random password and salt value, 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. Fix the rest of the $X$ indices and set $X_{z-4} = \mathcal{E}_{(20,\varphi_{20}(key))} \circ \mathcal{E}_{(16,\varphi_{16}(salt))}(C_z)$, $0 \leq z \leq 3$. Therefore, for any $0 \leq z \neq j \leq m\_cost - 1$

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

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

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

Since $z$ and $j$ are fixed integers, we have

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

Therefore, with probability at most $1 - 2^{-512}$ there are at least two distinct 128-bit elements in every set of five elements from the $X$ array. $\square$

**Claim 4.5.** *For any fixed random password and salt value, 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 $\ell$, $0 \leq \ell \leq m\_cost - \mu - 1$ and $\mu > 1$ such that there exists a leading sequence $X_0, X_1, \cdots, X_{\ell-1}$ called a *leader* and a *cycle* $X_\ell, X_{\ell+1}, \cdots, X_{\ell+\mu}$ of length $\mu$ such that $X_\ell = X_{\ell+\mu}$ [7].

Suppose $X = \{X_0, X_1, \cdots, X_{\mu-1}, X_0, X_1, \cdots, X_{\mu-1}, \cdots\}$ for some $\mu$ a power of two (since *m_cost* is a power of 2). Consider any two points $z$ and $j$ in distinct cycles such that $X_{z+k} = X_{j+k}$, $0 \leq k \leq 4$. We must have

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

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 $\mu \leq 4$. $\square$

Clearly, Claim 4.5 shows that $X$ does not contain any repeated cycle of length more than 4. This leads to Claim 4.6.

**Claim 4.6.** *For any fixed random password and salt value, 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 $\mu$ a power of two, then by Claim 4.5, $\mu = 2$ or 4.

If $\mu = 2$, $X_z = X_{z-2}$ for all $0 \leq z \leq m\_cost - 1$. Therefore, by the value of *m_cost*

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

a contradiction.

If $\mu = 4$, $X_z = X_{z-4}$ for all $0 \leq z \leq m\_cost - 1$. Therefore, by the value of $m\_cost$

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

a contradiction. □

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

**Claim 4.7.** *For any fixed random password and salt value, more than two repeated cycles in $X$ are unlikely.*

*Proof.* Suppose $\{X_z, X_{z+1}, \cdots, X_{z+\mu-1}\} = \{X_j, X_{j+1}, \cdots, X_{j+\mu-1}\} \subset X$, $0 \leq z \neq j \leq m\_cost - 1$ is a repeated cycle. Clearly,

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

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

Therefore, for a fixed random password and salt value, a repeated cycle occurs with probability at most $Pr^2[(X_{z-1} \oplus X_{z-4}) \oplus (X_{j-1} \oplus X_{j-4}) = z \oplus j] = 2^{-256}$. Hence, two repeated cycles have probability at most $2^{-512}$ which is unlikely for 384-bit passwords. □

Claim 4.7 implies that the adversary acquires no nontrivial complexity gain in exploiting regularities in the $X$ array.

## 4.2 Differential Properties of the Password Scheme

**Claim 4.8.** *Any set of 4 consecutive elements of the $X$ array for any two distinct passwords and a fixed salt value are distinct.*

*Proof.* We prove by contradiction.

Suppose there exist two distinct passwords and a fixed salt value such that

$$\{X^0_z, X^0_{z+1}, X^0_{z+2}, X^0_{z+3}\} = \{X^1_z, X^1_{z+1}, X^1_{z+2}, X^1_{z+3}\}$$

where $X^j$ is the [ordered] $X$ array for the $j^{th}$ password and $z \geq 0$.

Then, we have

$$\begin{aligned}\mathcal{E}_{(4,\vartheta_4(0))}(X^0_{z+2} \oplus X^0_{z-1} \oplus z+3) &= X^0_{z+3} \\ &= X^1_{z+3} \\ &= \mathcal{E}_{(4,\vartheta_4(0))}(X^1_{z+2} \oplus X^1_{z-1} \oplus z+3)\end{aligned}$$

which implies $X^0_{z-1} = X^1_{z-1}$. Similarly, we have $X^0_{z-2} = X^1_{z-2}$, $X^0_{z-3} = X^1_{z-3}$ and $X^0_{z-4} = X^1_{z-4}$.

Applying this iteratively, we arrive at $X^0_0 = X^1_0, X^0_1 = X^1_1, X^0_2 = X^1_2$ and $X^0_3 = X^1_3$. However, this can only happen with probability $2^{-512}$ a contradiction for 384-bit passwords by Claim 4.1 and Claim 4.2. □

**Claim 4.9.** *Related-password and related-salt attacks in M3dcrypt are unlikely.*

*Proof.* Follows from Claim 2.4, Claim 4.2, Claim 4.8 and the differential propagation ratio for $\mathcal{E}_{(4,\vartheta(0))}$ [12]. □

**Claim 4.10.** *For any fixed salt value, pairs of equivalent passwords in M3dcrypt are unlikely.*

*Proof.* Claim 2.3 shows that $\varphi_{20}(x) \neq \varphi_{20}(y)$ for any $x \neq y \in \mathbb{Z}_2^{384}$. On the other hand Claim 4.7 shows that for any two passwords $pd_0 \neq pd_1 \in \mathcal{K}_p$,

$$\Pi^3_{(X^0,t\_cost,tfactor)} \neq \Pi^3_{(X^1,t\_cost,tfactor)},$$

where $X^j = \chi(key_j, salt, m\_cost)$ and $key_j = pd_j||0^{384-|pd_j|}$.

Hence, by Claim 4.2,

$$\begin{aligned} &Pr\left[\mathcal{V}(pd_0, salt, m\_cost, tfactor) = \mathcal{V}(pd_1, salt, m\_cost, tfactor)\right] \\ &\qquad = Pr\left[\Pi^3_{(X^0,t\_cost,tfactor)}(0) = \Pi^3_{(X^1,t\_cost,tfactor)}(0)\right] \\ &\qquad = 2^{-384}. \end{aligned}$$

Therefore, pairs of equivalent keys are unlikely. □

## 4.3 Security of the M3dcrypt Password Scheme

**Theorem 4.1.** *Let $salt \in \mathbb{Z}_2^{256}$, $m\_cost \in \mathbb{Z}$ and $tfactor \in \mathbb{Z}$ be fixed, then the M3dcrypt password hashing function $M3dcrypt\_hash_{(salt,m\_cost,tfactor)} : \mathcal{K}_p \to \mathbb{Z}_2^{512}$ satisfies*

$$Adv^{owf}_{M3dcrypt\_hash_{(salt,m\_cost,tfactor)}}(t) \quad \leq \quad Adv^{prp}_F(8, t + O(640 + T_F)) + \frac{57}{2^{129}}$$

*where $F = \mathcal{E}_{(20,\mathcal{V}(.,salt,m\_cost,tfactor))}(.) : \mathcal{K}_p \times \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ and $T_F$ is time for a single iteration of $F$.*

*Proof.* For brevity, let

$$\begin{aligned} h(k) &= \text{M3dcrypt\_hash}_{(salt,m\_cost,tfactor)}(k) \\ &= \left(F(k, C_0), F(k, C_1), F(k, C_2), F(k, C_3)\right), \end{aligned}$$

where $F(k, x) = \mathcal{E}_{(20,\mathcal{V}(k,salt,m\_cost,tfactor))}(x)$ for all $k \in \mathcal{K}_p$ and all $x \in \mathbb{Z}_2^{128}$.

For any inverter $I$ for $h$, define

$$Adv^{owf}_{h,I}(t) = Pr\left[h(k^{'}) = y;\ k \xleftarrow{R} \mathcal{K}_p;\ y = h(k);\ k^{'} = I(y)\right]$$

where $I$ runs in time at most $t$ [6].

Clearly, for any inverter $I$ of $h$, we can construct a prf-adversary $A$ for $F$ as follows.

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

Since $A$ has oracle access to the function instance $f$ of either $F$ or $\text{Rand}^{128\to128}$ it can compute $y = f^2(x)$ for all $x \in \mathbb{Z}_2^{128}$. Therefore, it can run $I$ as a subroutine which recovers the key with probability $Adv^{owf}_{h,I}(t)$ whenever $f$ is an instance of $F$ and where $t$ is the maximum running time for $I$.

Moreover, since $h$ is a public function, $A$ can compute $h(k^{'})$ to confirm the result [4].

Therefore,

$$\begin{aligned}
Pr[f \xleftarrow{R} F : A^f = 1] &= Adv^{owf}_{h,I}(t) \\
Pr[f \xleftarrow{R} \text{Rand}^{128\to128} : A^f = 1] &\leq \frac{1.00002}{2^{383}}.
\end{aligned}$$

For the last inequality, we note the following. Given $k \in \mathcal{K}_p$ and any random $k^{'} \in \mathcal{K}_p$,

$$\begin{aligned}
Pr[h(k) = h(k^{'})] &= Pr[h(k) = h(k^{'}) \mid k = k^{'}] \cdot Pr[k = k^{'}] \\
&\quad + Pr[h(k) = h(k^{'}) \mid k \neq k^{'}] \cdot Pr[k \neq k^{'}] \\
&= 1 \cdot \frac{1}{2^{384}} + (1 - \frac{1}{2^{384}}) \cdot Pr[h(k) = h(k^{'}) \mid k \neq k^{'}] \\
&\leq \frac{1}{2^{384}} + Pr\left[h(k) = h(k^{'}) \mid \mathcal{V}(k) \neq \mathcal{V}(k^{'})\right] \\
&\quad \cdot Pr\left[\mathcal{V}(k) \neq \mathcal{V}(k^{'}) \mid k \neq k^{'}\right] \\
&\quad + Pr\left[h(k) = h(k^{'}) \mid \mathcal{V}(k) = \mathcal{V}(k^{'})\right] \\
&\quad \cdot Pr\left[\mathcal{V}(k) = \mathcal{V}(k^{'}) \mid k \neq k^{'}\right]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2^{384}} + \frac{1}{2^{512}} \cdot (1 - \frac{1}{2^{384}}) + 1 \cdot \frac{1}{2^{384}} \\
&\leq \frac{1}{2^{512}} + \frac{1}{2^{383}} \\
&\leq \frac{1.00002}{2^{383}}.
\end{aligned}
$$

We must have,

$$
\begin{aligned}
Adv_F^{prf}(A) &= Pr[f \xleftarrow{R} F : A^f = 1] - Pr[f \xleftarrow{\text{Rand}} {}^{128 \to 128} : A^f = 1] \\
&\geq Adv_{h,I}^{owf}(t) - \frac{1.00002}{2^{383}}.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
Adv_F^{prf}(8, t^{'}) + \frac{1.00002}{2^{383}} &\geq \max_I \{Adv_{h,I}^{owf}(t)\} \\
&= Adv_h^{owf}(t).
\end{aligned}
$$

Hence, by Proposition 2.5 of [4],

$$
Adv^{owf}_{\text{M3dcrypt\_hash}_{(salt, m\_cost, tfactor)}}(t) \leq Adv_F^{prp}(8, t^{'}) + \frac{57}{2^{129}},
$$

where $q = 8$ and $t^{'} = t + O(128 + 128 + 384 + T_F) = t + O(640 + T_F)$ [4]. $\square$

Theorem 4.1 shows that the M3dcrypt passowd scheme is a secure password hashing function as long as $\mathcal{E}_{(20, \mathcal{V}(., salt, m\_cost, tfactor))}(.) : \mathcal{K}_p \times \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128}$ is a secure PRP.

# 5 Efficiency analysis

## 5.1 Software Implementations

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 [5, 1].

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). In comparison, at creation in 1977, *crypt* could be evaluated about 3.6 times per second on a VAX-11/780 [14].

### 5.2 Hardware Implementations

The availability of large random access memory (RAM) in software implementations 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 customisation (e.g. external pipelining and/or other extensive parallelism) are contingent on the availability and cost of RAM [9].

However, by Claims 4.3, 4.4 and 4.7, 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, $0 \leq k \leq m\_cost - 1$.

In particular, any values $X_k$ in step (c) of Algorithm 3 not in memory will either have to be computed from scratch or from some point further down (in RAM) the computation chain.

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

## 6 Conclusion

We have described a new password hashing function which is secure as long as $\mathcal{E}_{(20,\mathcal{V}(.,salt,m\_cost,tfactor))}(.)$ is a secure PRP.

## References


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

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

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

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

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

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

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

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

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

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