مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (31) العدد (3) 2009
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (31) No. (3) 2009

تقوية لخوارزمية التشفير الصغيرة جداً

ساري حاج حسين *

(تاريخ الإيداع 21 / 4 / 2009. قُبِل للنشر في 2009/7/8)

□ ملخّص □

صمم العالمان David Wheeler و Roger Needham خوارزمية التشفير الصغيرة جداً Tiny Encryption Algorithm (TEA) في مخبر الحاسوب التابع لجامعة كامبردج، وتم نشرها لأول مرة في ورشة عمل تشفير البرمجيات السريع Fast Software Encryption Workshop في العام 1994.

تجمع هذه الخوارزمية خواص الأمان وسهولة التضمين وعدم الحاجة إلى جداول مخصصة والأداء المعقول، وهي صغيرة لدرجة تسمح بتضمينها في أي برنامج وعلى أي حاسوب.

نقدم في هذا البحث دراسة لهذه الخوارزمية موضحين نقاط الضعف فيها والتعديلات التي دخلت عليها منذ ظهورها وحتى الآن، والتي رغم كثرتها لم تنفِ الحاجة المستمرة لتقوية هذه الخوارزمية. نقترح بعد هذه الدراسة أسلوباً جديداً لتقوية هذه الخوارزمية، مبرهنين على نجاعته وسرعته مقارنةً بأساليب التقوية الأخرى.

الكلمات المفتاحية: تشفير فيستل، خوارزمية التشفير الصغيرة جداً، تحليل الشفرة التفاضلي، النسبة الذهبية، الهجوم المنزلق، الخوارزميات الجينية، الهجوم التفاضلي المستحيل، الهجوم التفاضلي الناقص، هجوم المفاتيح المتساوية، هجوم المفتاح المتصل.

* قائم بالأعمال معاون–قسم هندسة البرمجيات ونظم المعلومات–كلية الهندسة المعلوماتية–جامعة تشرين – اللاذقية – سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (31) العدد (3) 2009
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (31) No. (3) 2009

Strengthening of The Tiny Encryption Algorithm (TEA)

Sari Haj Hussein *

(Received 21 / 4 / 2009. Accepted 8 / 7 / 2009)

□ ABSTRACT □

The Tiny Encryption Algorithm (TEA) was designed by David Wheeler and Roger Needham of the Cambridge Computer Laboratory, and first presented at the Fast Software Encryption workshop in 1994.

This algorithm is a compromise for safety, ease of implementation, lack of specialized tables, and reasonable performance, and is short enough to integrate into almost any program on any computer.

In this paper, we introduce a study of this algorithm highlighting its drawbacks and the modifications to it since it was suggested and until now. These modifications, although many, did not eliminate the continuous need to strengthen this algorithm. After this study, we propose a new method for strengthening TEA proving its efficiency and speed of performance when compared to other strengthening methods.

Keywords: Feistel Cipher, Tiny Encryption Algorithm (TEA), Differential Cryptanalysis, Golden Ratio, Slide Attack, Genetic Algorithms, Impossible Differential Attack, Truncated Differential Attack, Equivalent Keys Attacks, Related-Key Attack.

* **Assistant Work Doer-Department of Software Engineering and System Analysis-Faculty of Information Engineering-Tishreen University – Lattakia - Syria**

مقدمة:

تزداد أهمية أمن المعلومات هذه الأيام باطراد مع ازدياد انتشار وتعقيد أنظمة الحواسيب وتشكل خوارزميات وبروتوكولات التشفير المكون المركزي في الأنظمة التي تحمي البيانات المنقولة عبر الشبكات المختلفة. إن أمن هذه الأنظمة يعتمد بشكل كبير على الطرق المتبعة لإدارة وتأسيس وتوزيع المفاتيح التي توظفها تقنيات التشفير المختلفة. حتى لو كانت خوارزمية التشفير مثالية من الناحيتين النظرية والتطبيقية فإن قوتها ستكون عديمة النفع فيما لو تعاملنا مع المفاتيح بشكل خاطئ.

أهمية البحث وأهدافه:

تأتي أهمية هذا البحث من الأهمية المتزايدة لمسألة الأمان، نظراً للنمو السريع والاستخدام الواسع للبيانات الرقمية، إضافة إلى العدد الهائل من الأعمال التجارية التي يتم إنجازها على شبكة الإنترنت، والبروز المرعب لظاهرة الإرهاب العالمي، والتي غذت بمجموعها الحاجة لطرق أفضل لحماية هذه الحواسيب والمعلومات التي تخزنها وتعالجها وتنقلها، وأدت في نهاية المطاف إلى نشوء منظمات متخصصة كبيرة الحجم تشترك كلها في هدف واحد ألا وهو حماية أمن وموثوقية الأنظمة المعلوماتية...

أما أهداف البحث فتتلخص في محاولة تقوية الخوارزمية TEA التي لم تعد تتمتع بمستوى مقبول من الأمان نتيجة تعرضها للعديد من محاولات تحليل الشفرة الناجحة.

طرائق البحث ومواده:

يبدأ هذا البحث بشرح مفصل للخوارزمية TEA ولروتيني التشفير وفك التشفير، ثم يقارن بين أداء هذه الخوارزمية وبعض خوارزميات التشفير الأخرى، مبيناً الكفاءة العالية التي تتمتع بها هذه الخوارزمية.

بعدها يشرح البحث وبشكل ملخص محاولات تحليل الشفرة التي تعرضت لها الخوارزمية TEA منذ ظهورها وحتى الآن، وأخيراً يطرح البحث أسلوب التقوية المقترح مبيناً مدى قوته وسرعته مقارنة بالخوارزمية TEA.

أُجري البحث على شبكة الإنترنت مطلع العام 2007 أثناء عملنا عن بعد مع فريق من المطورين على أحد المشاريع مفتوحة المصدر في موقع استضافة البرمجيات المجانية http://www.tigris.org.

أما بالنسبة لأدوات البحث فقد استخدمنا الأدوات البرمجية التالية:

1) لغة البرمجة C من أجل التضمين البرمجي لروتينات التشفير وفك التشفير.
2) البرنامج ENT لاختبار مدى جودة أسلوب التقوية المقترح.

دراسات مرجعية:

إن للتشفير تاريخاً طويلاً، إذ إنه منذ نشوء اللغات المحكية والمكتوبة حاول الناس إخفاء معاني الكلمات؛ أي أنهم وظفوا تقنيات التشفير بشكل أو بآخر. وعلى الرغم من كون الرياضيات القابعة وراء هذه التقنيات بدائية للغاية إلا أنها ما تزال تعتبر أنظمة تشفير [1].

حتى الحرب العالمية الثانية كان ينظر إلى التشفير Cryptography على أنه فن (أكثر من كونه علماً) إذ كان يستخدم بشكل رئيسي في القوات المسلحة. حول الإنجازان العلميان التاليان التشفير من فن إلى علم:

1) أثناء الحرب العالمية الثانية، طور العالم Claude Shannon نظرية رياضية في الاتصالات ونظرية أخرى للاتصالات في الأنظمة السرية، عندما كان يعمل في مختبرات AT&T. بعد نشر هاتين النظريتين، ولد فرع جديد من الأبحاث عرف باسم نظرية المعلومات Information Theory.

2) قدم العالمان Whitfield Diffie و Martin Hellman فكرة تشفير المفتاح العام في جامعة ستانفورد Stanford University في العام 1970. وقد توجا عملهما بتطوير بروتوكول اتفاقية مفتاح Key Agreement Protocol الذي يسمح لطرفين لا يشتركان بأي سر مسبق بأن يتبادلا بعض الرسائل عبر قناة عامة وأن يؤسسا مفتاحاً سرياً مشتركاً.

ما هو التشفير؟

وفقاً للمستند RFC 2828 [2] فإن التشفير Cryptography هو ذلك العلم الرياضي الذي يحول البيانات ويعالج معناها، بشكل لا يمكن فهمه، ويمنع التبديل غير الملحوظ أو الاستخدام غير المحول. وإذا كان هذا التحويل عكوساً فإن التشفير يتعامل أيضاً مع استعادة البيانات المشفرة إلى شكلها المفهوم.

أما تحليل الشفرة Cryptanalysis فهو ذلك العلم الرياضي الذي يتعامل مع تحليل نظام التشفير بغرض اكتساب المعرفة اللازمة لكسر أو الدوران حول الحماية التي صمم هذا النظام ليقدمها. وبالتالي فإن محلل الشفرة Cryptanalysis هو خصم للمشفر Cryptographer.

وتصنف أنظمة التشفير عادةً إلى [1]:

1) أنظمة التشفير بلا مفتاح Unkeyed Cryptographic Systems: نظام التشفير بلا مفتاح هو نظام تشفير لا يستخدم أية معاملات سرية.

2) أنظمة تشفير المفتاح السري Secret Key Cryptographic Systems أو أنظمة التشفير المتناظرة Symmetric Cryptographic Systems: نظام التشفير المتناظر هو نظام تشفير يستخدم نفس المفتاح للتشفير وفك التشفير.

3) أنظمة تشفير المفتاح العام Public Key Cryptographic Systems أو أنظمة التشفير غير المتناظرة Asymmetric Cryptographic Systems: نظام التشفير غير المتناظر هو نظام تشفير يستخدم مفتاحين مختلفين للتشفير وفك التشفير، وفي الحالة المثالية يكون من غير الممكن حساب مفتاح فك التشفير بالاعتماد على مفتاح التشفير.

أما أنظمة التشفير المتناظرة فتصنف إلى [1]:

1) التشفير الدفقي Stream Cipher: الذي يشفر بتاً واحداً من النص الأصلي في كل مرة.

2) التشفير الكتلي Block Cipher: الذي يأخذ عدداً من البتات (64 بتاً مثلاً) ويشفرها معاً كوحدة واحدة.

إن التشفير المتناظر هو عنصر رئيسي في العديد من أنظمة الاتصالات السرية وقد تم اختراع وتضمين العديد من خوارزمياته في التجهيزات وفي البرمجيات.

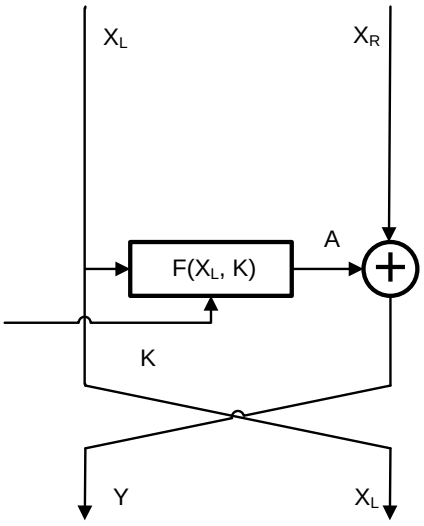
الترميز المستخدم

- سنلحق بالأعداد الست عشرية المحرف السفلي h كما يلي: $10_h = 16$.

- سنرمز لإزاحة x إلى اليسار بمقدار y بت بالرمز x << y في حين سنرمز لإزاحة x إلى اليمين بمقدار y بت بالرمز x >> y.
- سنرمز لدوران x إلى اليسار بمقدار y بت بالرمز x <<< y في حين سنرمز لدوران x إلي اليمين بمقدار y بت بالرمز x >>> y.
- سنرمز إلى العملية المنطقية أو الحصرية Exclusive-OR لـ x مع y بالرمز $x \oplus y$.
- نسمي المجموعة {0, 1, 2, .., n – 1} بمجموعة الأعداد الصحيحة بالمقياس n ونرمز لها بالرمز Z_n وتتم كل عمليات الجمع والطرح والضرب في Z_n بالمقياس n أي modulo n.

مثلاً: Z_{25} = {0, 1, 2, .., 24} وفيها يكون 4 = 16 + 13 لأن 4 = 25 mod 29 = 16 + 13، وبالمثل فإن 8 = 25 mod 208 = 13.16.

- ليكن $x, y \in Z_{2^n}$ فإننا نرمز للعملية $(x + y) \bmod 2^n$ والتي تمثل جمع x مع y في Z_{2^n} بالرمز $x \boxplus y$.
- ليكن $x, y \in Z_{2^n}$ فإننا نرمز للعملية $(x - y) \bmod 2^n$ والتي تمثل طرح y من x في Z_{2^n} بالرمز $x \boxminus y$.
- تشفير فيستل Feistel Cipher [3] هو نوع من أنواع التشفير الكتلي التكراري، تقسم فيه كتلة النص الأصلي X إلى كتليتين متساويتي الطول X_L, X_R ويطبق تابع الجولة أو Round Function ويرمز F على إحدى هاتين الكتليتين باستخدام مفتاح فرعي K، ثم يخضع خرج التابع F إلى عملية أو حصرية مع الكتلة الثانية. بعدها تُستبدل الكتلتان. تستخدم كل جولة تالية نفس الأسلوب تماماً، باستثناء الجولة الأخيرة التي لا تتضمن عملية استبدال. يوضح الشكل (1) ذلك:



الشكل (1): جولة في تشفير فيستل

خوارزمية التشفير الصغيرة جداً

صمم العالمان David Wheeler و Roger Needham الخوارزمية TEA في مخبر الحاسوب التابع لجامعة كامبردج، وتم نشرها لأول مرة في ورشة عمل تشفير البرمجيات السريع Fast Software Encryption Workshop في العام 1994 [4].

تجمع هذه الخوارزمية خواص الأمان وسهولة التضمين وعدم الحاجة إلى جداول مخصصة والأداء المعقول وهي صغيرة لدرجة تسمح بتضمينها في أي برنامج وعلى أي حاسوب.

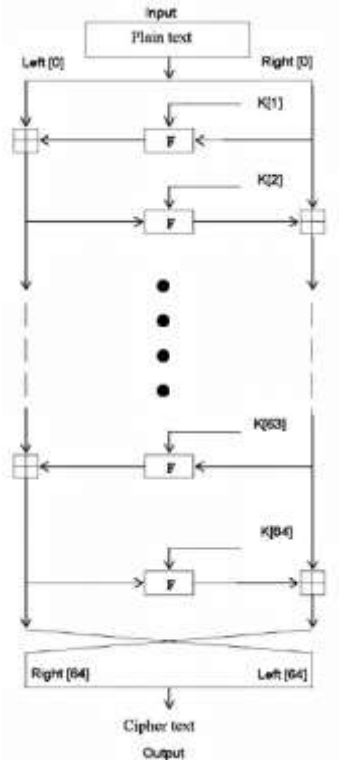
إن هذه الخوارزمية ما هي إلا نوع من أنواع تشفير فيستل، يستخدم عمليات جبرية متنوعة مستقلة إحصائياً. فهي تعمل على كتل مؤلفة من 64 بتاً وتستخدم مفتاحاً بطول 128 بتاً ولها 64 جولة.

تسبب الإزاحة المضاعفة في TEA دمج كل بتات البيانات والمفتاح بشكل متكرر. وخوارزمية جدولة المفتاح Key Schedule Algorithm بسيطة للغاية، إذ يقسم المفتاح K المؤلف من 128 بتاً إلى أربع كتل كل منها مؤلف من 32 بت (K[0], K[1], K[2], K[3]) = K.

يبدو بأن هذه الخوارزمية تقاوم بفعالية عالية تحليل الشفرة التفاضلي Differential Cryptanalysis [5] وتحقق نشراً كاملاً Diffusion (حيث أن اختلاف بت واحد في النص الأصلي سيحدث اختلافات في 32 بتاً تقريباً في النص المشفر) كما أن توفيرها في الوقت على محطات العمل مثير للإعجاب بالفعل.

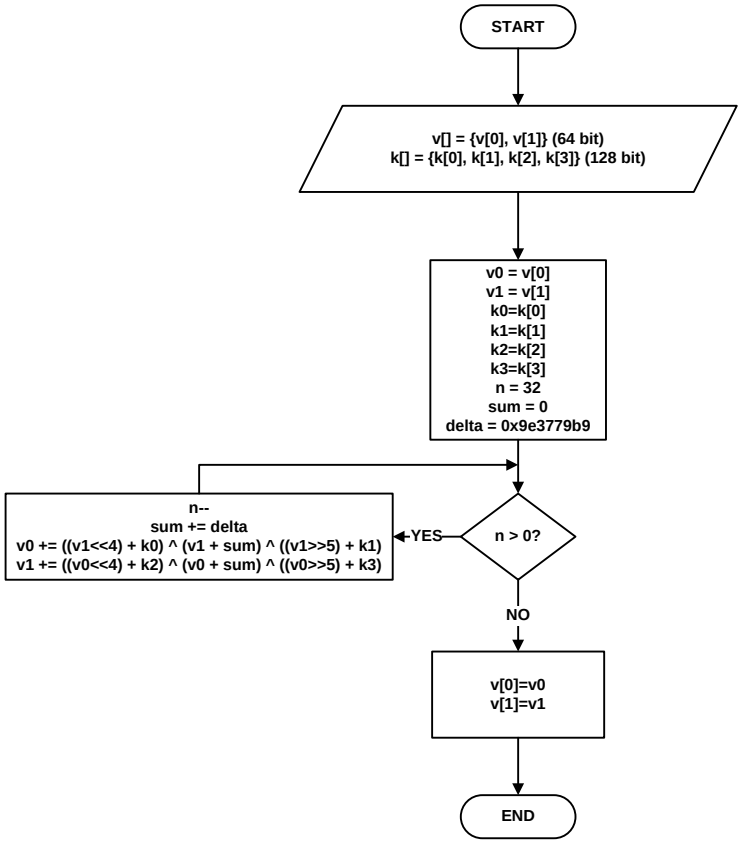
■ **التشفير**

يبين الشكل (2) البنية المجردة لطريقة التشفير في TEA:



الشكل (2): البنية المجردة لطريقة التشفير في TEA [6]

أما الشكل (3) فيمثل المخطط التدفقي لطريقة التشفير في TEA:



الشكل (3): المخطط التدفقي لطريقة التشفير في TEA [7]

في حين يتضمن الشكل (4) كود التشفير في TEA مكتوباً بلغة C مع التنويه إلى أن البيانات الأصلية مخزنة في المصفوفة []v والمفتاح مخزن في المصفوفة []k أما نتيجة التشفير (البيانات المشفرة) فستخزن في المصفوفة []v أيضاً بعد استبدال البيانات الأصلية [8]:

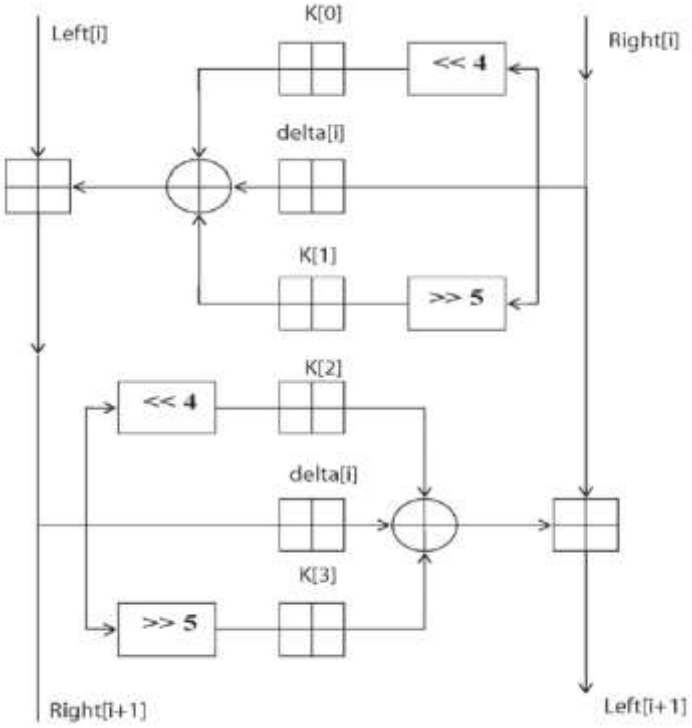
```
void encrypt(unsigned long* v, unsigned long* k) {
     unsigned long v0=v[0], v1=v[1], sum=0; /* set up */
     unsigned long delta=0x9e3779b9, n = 32; /* a key schedule constant */
     unsigned long k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */
     while (n-->0) { /* basic cycle start */
          sum += delta;
          v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
          v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3); /* end cycle */
     }
     v[0]=v0; v[1]=v1;
}
```

الشكل (4): كود التشفير في TEA مكتوباً بلغة C

إن دخل الخوارزمية هو كتلة النص الأصلي، والمفتاح K. يكون النص الأصلي P = (Left[0], Right[0]) والنص المشفر C = (Left[64], Right[64]). تقسم كتلة النص الأصلي المؤلفة من 64 بتاً إلى نصفين Left[0] و Right[0] كل منهما مؤلف من 32 بتاً. يستخدم كل نصف لتشفير الآخر عبر 64 جولة معالجة، ثم يدمجا معاً لتوليد كتلة النص المشفر.

- إن دخل كل جولة i هو Left[i – 1] و Right[i – 1] وهما مشتقان من الجولة السابقة، بالإضافة إلى مفتاح فرعي K[i – 1] مشتق من المفتاح الكلي K المؤلف من 128 بت.
- إن المفاتيح الفرعية K[i] مختلفة عن بعضها البعض وعن المفتاح K.
- إن الثابت $delta = (\sqrt{5}-1)*2^{31} = 9E3779B9_h$ مأخوذ من النسبة الذهبية Golden Ratio ويستخدم للتأكد من أن المفاتيح الفرعية متمايزة وقيمته الفعلية لا أهمية لها في التشفير.
- يختلف تابع الجولة قليلاً عنه في تشفير فيستل التقليدي، إذ يستخدم الجمع بالمقياس 2^{32} كمعامل للدمج بدلاً من أو الحصرية.

يبين الشكل (5) التفاصيل الداخلية للدورة i من TEA (الدورة تعادل جولتين).



الشكل (5): تجريد للدورة i من TEA [6]

يتألف تابع الجولة F من إضافة المفتاح وعملية أو حصرية وعمليتي إزاحة يمينية ويسارية. ويمكن وصف الخرج (Left[i+1], Right[i+1]) للدورة i من TEA مع الدخل (Left[i], Right[i]) كما يلي:

Left [i+1] ⊞Left[i] F(Right[i], K [0, 1], delta[i]),
Right [i +1] ⊞Right[i] F(Right[i +1], K [2, 3], delta[i]),
delta[i] = (i +1)/2 * delta,

ويعرف التابع F كما يلي:

F(M, K[j,k], delta[i]) = ⊞(M << 4) ⊞K[j]) ⊕ (M delta[i⊞) ⊕ ((M >> 5) K[k]).

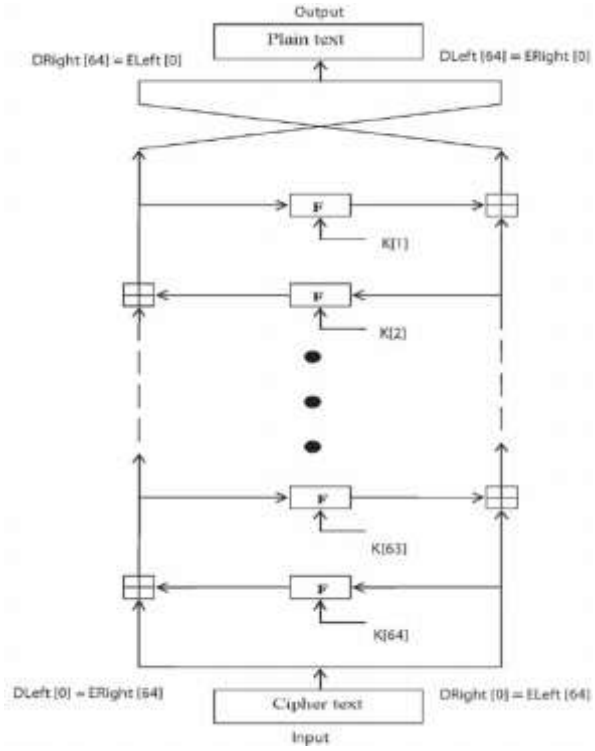
إن لتابع الجولة F نفس البنية العامة لكنه يأخذ مفتاحاً فرعياً مختلفاً K[i] في كل جولة.

إن خوارزمية جدولة المفتاح بسيطة للغاية؛ إذ يقسم المفتاح K المؤلف من 128 بتاً إلى أربع كتل كل منها مؤلف من 32 بتاً K = (K[0], K[1], K[2], K[3]). ويستخدم المفتاحان K[0] و K[1] في الجولات الفردية في حين يستخدم المفتاحان K[2] و K[3] في الجولات الزوجية.

■ فك التشفير

إن عملية فك التشفير مماثلة لعملية التشفير؛ إذ يستخدم النص المشفر كدخل للخوارزمية، وتستخدم المفاتيح الفرعية K[i] بطريقة معكوسة.

يبين الشكل (6) البنية المجردة لطريقة فك التشفير في TEA:



الشكل (6): البنية المجردة لطريقة فك التشفير في TEA [6]

إن المخطط التدفقي لطريقة فك التشفير في TEA مماثل تقريباً للمخطط التدفقي لطريقة التشفير في TEA الذي ورد في الشكل (3) باستثناء أن نهيئ المتحول sum على القيمة delta << 5 ونستبدل كل عملية =+ بعملية =-.

يتضمن الشكل (7) كود فك التشفير في TEA مكتوباً بلغة C، مع التنويه إلى أن البيانات المشفرة مخزنة في المصفوفة []v والمفتاح مخزن في المصفوفة []k أما نتيجة فك التشفير (البيانات الأصلية) فستخزن في المصفوفة []v أيضاً بعد استبدال البيانات المشفرة [8]:

```
void decrypt(unsigned long* v, unsigned long* k) {
     unsigned long v0=v[0], v1=v[1], sum=0xC6EF3720, i;
     /* set up, sum=delta<<5 */
     unsigned long delta=0x9e3779b9, n = 32; /* a key schedule constant */
     unsigned long k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */
```

```
    while (n-->0) { /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta; /* end cycle */
    }
    v[0]=v0; v[1]=v1;
}
```

الشكل (7): كود فك التشفير في TEA مكتوباً بلغة C

لاحظ أن القيمة الوسيطة في عملية فك التشفير مماثلة للقيمة الموافقة في عملية التشفير بعد استبدال نصفي القيمة. مثلاً إذا كان خرج جولة التشفير n كما يلي:

ELeft[i] || ERight[i] (ELeft[i] concatenated with ERight[i]).

فإن دخل جولة فك التشفير 64i الموافقة هو:

DRight[i] || DLeft[i] (DRight[i] concatenated with DLeft[i]).

بعد التكرار الأخير في عملية التشفير، فإن نصفي الخرج يستبدلان بحيث يصبح النص المشفر ERight[64] ELeft[64] || ويكون خرج هذه الجولة هو النص المشفر النهائي C. لاحظ هنا بأن هذا النص المشفر يستخدم كدخل لعملية فك التشفير ويكون دخل الجولة الأولى ELeft[64] || ERight[64] وهو مساوٍ للاستبدال ذي 32 بتاً لخرج الجولة 64 من عملية التشفير.

مقارنة بين أداء TEA وأداء بعض الخوارزميات الأخرى

يقارن الجدول (1) بين أداء TEA وأداء بعض الخوارزميات من حيث طول المفتاح والسرعة المقدرة:

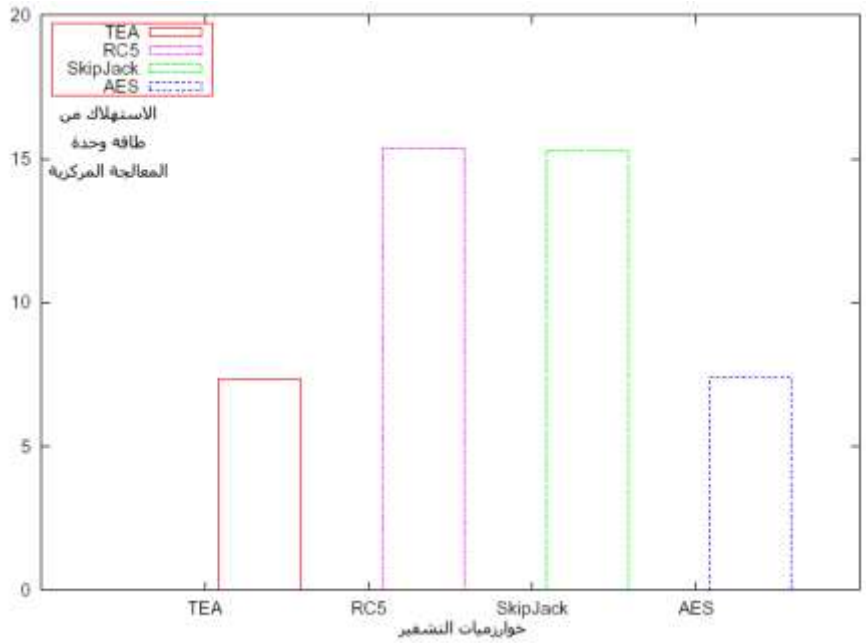
الجدول (1): أداء TEA بالمقارنة مع أداء بعض الخوارزميات الشائعة الأخرى [9]

الخوارزمية	طول المفتاح/طول قيمة التجزئة (بالبتات)	السرعة المقدرة (بالكيلوبايت في الثانية)
TEA	128	700
DES	56	350
Triple-DES	112	120
IDEA	128	700
RSA	512	7
RSA	2048	1
MD5	128	1740
SHA	160	750

لاحظ الأداء المذهل الذي تقدمه TEA.

مقارنة بين استهلاك TEA من طاقة وحدة المعالجة المركزية واستهلاك بعض الخوارزميات الأخرى

يقارن الشكل (8) بين استهلاك TEA واستهلاك بعض الخوارزميات الأخرى الشائعة من طاقة وحدة المعالجة المركزية:



الشكل (8): استهلاك TEA من طاقة وحدة المعالجة المركزية بالمقارنة مع استهلاك بعض الخوارزميات الشائعة الأخرى [10]

لاحظ أيضاً روعة ما تقدمه TEA.

تحليل الشفرة

نتطرق الآن إلى مجموعة من هجمات تحليل الشفرة على الخوارزمية TEA لنخلص في نهاية المطاف إلى أن هذه الخوارزمية لا تؤمن للأسف درجة الأمان المرجوة.

■ هجمات المفاتيح المتساوية والمفاتيح المتصلة على TEA

استهدف الهجوم الأول على TEA جدولة مفتاحها البسيطة للغاية إذ إن TEA تقسم المفتاح K المؤلف من 128 بتاً إلى أربع كتل كل منها مؤلف من 32 بتاً (K = (K[0], K[1], K[2], K[3]. ويستخدم المفتاحان K[0] و K[1] في الجولات الفردية في حين يستخدم المفتاحان K[2] و K[3] في الجولات الزوجية، كما ذكرنا في سابقاً.

في كل جولة يُستخدم مضاعف مختلف للثابت delta الأمر الذي يجعل عملية التحويل مختلفة تماماً. دون هذا المضاعف تصبح TEA عرضةً للهجوم المنزلق Slide Attack الذي وصفه لأول مرة العالمان Biryukov وWagner في العام 1999 [11] ولاحظ العالم Fleming إمكانية حدوثه في TEA في العام 1996 [12].

تعاني TEA من تساوي المفاتيح [13] إذ إن العكس المستمر للبتات الأكثر أهمية في K[0] و K[1] لا يؤثر على التشفير لأنها تلغي بعضها. لنشرح ذلك بالتفصيل...

القضية 1: لتكن a كلمة مؤلفة من n بت فإن عكس البت الأكثر أهمية في a يكافئ الإضافة النمطية أي:

$$a \oplus 2^{n-1} \equiv a \boxplus 2^{n-1}$$

القضية 2: لتكن a و b كلمتين مؤلفتين من n بت فإن المطابقة التالية صحيحة:

$$a \boxplus (b \oplus 2^{n-1}) \equiv a \boxplus (b \boxplus 2^{n-1}) \equiv (a \boxplus b) \boxplus 2^{n-1} \equiv (a \boxplus b) \oplus 2^{n-1}$$

القضية 3: لتكن $TEA_i(z, K_0, K_1)$ تشير إلى تابع الجولة i من TEA المطبق على كلمة z مؤلفة من 32 بتاً مع المفاتيح الفرعية K_0 و K_1 عندها:

$$\begin{aligned}
\mathrm{TEA}_i(z, K_0 \oplus 2^{31}, K_1 \oplus 2^{31}) &= [(z \ll 4) \boxplus (K_0 \oplus 2^{31})] \oplus \\
&\quad [(z \gg 5) \boxplus (K_1 \oplus 2^{31})] \oplus [z \boxplus \delta_i] \\
&= [(z \ll 4) \boxplus K_0] \oplus 2^{31} \oplus \\
&\quad [(z \gg 5) \boxplus K_1] \oplus 2^{31} \oplus [z \boxplus \delta_i] \\
&= [(z \ll 4) \boxplus K_0] \oplus \\
&\quad [(z \gg 5) \boxplus K_1] \oplus [z \boxplus \delta_i] \\
&= \mathrm{TEA}_i(z, K_0, K_1)
\end{aligned}$$

وبشكل مشابه، فإن العكس المستمر للبتات الأكثر أهمية في K[2] و K[3] لا يؤثر على التشفير.

نستنتج أن لكل مفتاح في TEA ثلاث مفاتيح مساوية له، الأول ينتج عن عكس البتات الأكثر أهمية في K[0] و K[1] والثاني ينتج عن عكس البتات الأكثر أهمية في K[2] و K[3] أما الثالث والأخير فينتج عن عكس البتات الأكثر أهمية في K[0] و K[1] و K[2] و K[3].

ومن ثم فإن فضاء المفتاح الفعال في TEA ينخفض من 2^{128} إلى 2^{126}. لذلك فإن استخدام TEA غير مناسب لحساب ملخص الرسالة Message Digest.

مثال من الحياة:

تستند عملية الإقلاع في جهاز الألعاب Microsoft's Xbox على TEA المضمنة كتابع تجزئة في ذاكرة الجهاز ROM. يُستخدم تابع التجزئة هذا للتحقق من أن محتويات الذاكرة الومضية Flash Memory سليمة، ولم يتم تعديلها أو العبث بها بطريقة ما.

حاول المبرمج البريطاني Andy Green مع جماعة من القراصنة المحترفين كسر قواعد الجهاز Xbox وذلك بمحاولة تشغيل النظام Linux عليه!

يقول Green "إن الذاكرة الومضية في الجهاز موقعة بملخص رسالة مشفر (وهو عبارة عن طريقة لتلخيص مقدار كبير من البيانات بسلسلة عشوائية قصيرة تكون شديدة الحساسية لأي تغيير يطرأ على البيانات الأصلية)".

إن الضعف المعروف في الخوارزمية TEA المستخدمة لحساب ملخص الرسالة في Xbox دفع Green للقول: "في كل مجموعة مؤلفة من 64 بتاً، لا يمكن للخوارزمية TEA أن تكشف التغيير عندما يتغير البتان 32 و 64 معاً".

نجح Green ورفاقه بالفعل في الاستفادة من هذا الضعف واستطاعوا التحكم الكامل بالجهاز Xbox.

يقول Green: "تبين لنا بأن عكس البتين 32 و 64 من الكتلة الأولى المؤلفة من 64 بتاً من المنطقة المحمية جعل Xbox يبدأ التنفيذ من الذاكرة RAM وبالتالي أصبح سهلاً علينا تضمين أمر في الذاكرة RAM يسمح للمستخدم بالتحكم بالجهاز" [14].

لقد تم نشر العديد من هجمات المفاتيح المتصلة على TEA [15] وأفضلها يحتاج إلى 2^{23} نصاً أصلياً مختارين مشفرين باستخدام مفتاحين متصلين ويستغرق 2^{32} زمن حساب.

إن هجمات المفاتيح المتساوية والمفاتيح المتصلة هذه، هي التي دفعت المصممين إلى البحث عن بدائل لهذه الخوارزمية.

■ **إيجاد مميزات للخوارزمية TEA**

تم القيام ببعض التجارب لإيجاد مميزات للخوارزمية TEA المؤلفة من عدد قليل من الجولات، وقد وجد العالم Hernandez في العام 2001 [16] أنه من الممكن إيجاد مميزات للخوارزمية TEA المؤلفة من 5 دورات أو أقل، وذلك باستخدام 2^{25} نصين أصليين مختارين.

في عدد من المحاولات الأخرى استخدم Hernandez [17, 18, 19] في الأعوام 2002 و 2003 الخوارزميات الجينية Genetic Algorithms لإيجاد مميزات لعدد أكبر من جولات TEA. تستند فكرته إلى تحديد مجموعة فرعية من فضاء النص الأصلي P تترافق مع قسم من فضاء النص المشفر C بطريقة متقطعة. تعرف المجموعة الفرعية من النص الأصلي بقناع البت M الذي يحدد بتات يجب أن تكون ثابتة على الصفر أي مجموعة:

$$P_M \subseteq P$$

معرفة كما يلي:

$$P_M = \{p \in P \mid (p\ AND\ M) = p\}$$

يعرف قسم النص المشفر بتنظيم نص مشفر إلى البتات 8 أو 10 الأقل أهمية من نصفه الأيسر المؤلف من 32 بتاً أي:

$$\pi(c) = (c\ AND\ 0x000000FF00000000)$$

يتم إنشاء قناع البت هذا باستخدام النظرية الجينية.

لقد ثبتت فعالية الأسلوب الذي اتبعه Hernandez في إيجاد مميزات لـ TEA المؤلفة من 4 دورات.

■ **الهجمات التفاضلية على TEA**

تعرضت TEA في السنوات القليلة المنصرمة إلى عدد كبير من الهجمات التفاضلية. فقد استخدم Moon في العام 2002 [20] الهجوم التفاضلي المستحيل Impossible Differential Attack للهجوم على TEA المؤلفة من 11 جولة وقد تطلب الأمر $2^{52.5}$ نصاً أصلياً مختارين وبتعقيد زمني 2^{84}.

كما تم تحليل TEA بحث عن نقاط الضعف إزاء الهجمات التفاضلية والتفاضلية الناقصة Truncated Differential من قبل العالم Hong في العام 2003 [21, 22] وقد وجد Hong أن الهجوم التفاضلي الناقص مع احتمال 1 يمكنه كسر 17 جولة من TEA باستخدام 1920 نصاً أصلياً مختاراً وبتعقيد زمني $2^{123.37}$.

■ **ملخص الهجمات**

يبين الجدول (2) ملخصين بالهجمات التي تعرضت لها TEA مع خواص هذه الهجمات:

الجدول (2): الهجمات التي تعرضت لها TEA

نوع الهجوم	الجولات	النص الأصلي	الزمن	من قام به؟ ومتى؟
Equivalent Keys	Any	1	2^{126}	Kelsey في العام 1996
Related-Key	64	2^{23}	2^{32}	Kelsey في العام 1997
SAC Distinguisher	10	2^{25}	-	Hernandez في العام 2001
Partitioning	2-8	-	-	Hernandez في العام 2003
Impossible Differential	11	$2^{52.5}$	2^{84}	Moon في العام 2002

أسلوب التقوية المقترح

من المعلوم أن TEA قد كسرت اليوم كما تم اكتشاف العديد من نقاط الضعف فيها، ومعظم نقاط الضعف هذه ناتجة عن جدولة المفتاح. بالنتيجة قام مؤلفا TEA وهما Roger Needham و David Wheeler بتقديم نسخ موسعة من هذه الخوارزمية، جدولة المفتاح فيها أعقد لكن هذه النسخ الموسعة لم تحظَ للأسف بالقبول الذي حظيت به TEA ربما لأنها أبطأ.

إن الضعف المعروف في الخوارزمية TEA إضافة إلى شيوع استخدامها دفعنا إلى التفكير بأن الوقت قد حان بالفعل لتقوية هذه الخوارزمية تقوية ناجعة. يجب أن يتمتع أسلوب التقوية بما يلي:

- يجب أن تكون سرعته مساوية لسرعة الخوارزمية TEA كي نتمكن من استخدامه في نفس التطبيقات في الحياة العملية.
- يجب أن يكون أكثر قوة بحيث لا يعاني من الضعف الموجود في الخوارزمية TEA.

ولتطوير أسلوب التقوية استخدمنا تقنيات البرمجة الجينية، وهي من التقنيات القوية جداً في الوقت الراهن.

■ وصف أسلوب التقوية المقترح

إن أسلوب التقوية المقترح يستهدف جدولة المفتاح في الخوارزمية TEA فهو أكثر تعقيداً، لكنه يبقى بسيطاً بحيث لا ينهك زمن تنفيذ التشفير. لزيادة مستوى العشوائية في جدولة المفتاح جعلنا خرج كل جولة يغذي الموقع i%4 من مصفوفة المفتاح، الأمر الذي يجعل مصفوفة المفتاح متغيرة مع تقدم تنفيذ الخوارزمية، بحيث يتم توليد مفاتيح جولة جديدة لكل دورة باستخدام مصفوفة المفتاح الأصلية المؤلفة من 128 بتاً لنحصل في النهاية على جدولة مفتاح تقارب في طبيعتها مولدات الأعداد العشوائية!

نرى في الشكل (9) كود التشفير المقوى مكتوباً بلغة C:

```
void strengthen_encrypt(unsigned long *data,unsigned long *result,unsigned long *key)
{
unsigned long b0=data[0], b1=data[1],i,k[4]={key[0],key[1],key[2],key[3]}, sk;
for(i=0; i< 16; i++)
{
     sk=k[i%4]=((k[0]+k[1])+((k[2]+k[3])^(k[0]<<k[2])));
     b0 +=((sk+b1)<<9)^((sk-b1)^((sk+b1)>>14));
     b1 +=((sk+b0)<<9)^((sk-b0)^((sk+b0)>>14));
}
result[0]=b0;
result[1]=b1;
}
```

الشكل (9): كود التشفير المقوى مكتوباً بلغة C

يستقبل الإجراء بيانات النص الأصلي في المعامل data ويضع النص المشفر في المعامل result أما key فنخزن فيه المفتاح المؤلف من 128 بتاً.

قلنا سابقاً إن الخوارزمية TEA تتبع تشفير فيستل، لذلك فإن فك التشفير يتم بنفس الطريقة التي تم بها التشفير ولكن مع تطبيق مفاتيح الجولة بترتيب عكسي.

نرى في الشكل (10) كود فك التشفير الموافق مكتوباً بلغة C:

```
void strengthen_decrypt(unsigned long *data,unsigned long *result,unsigned long
*key)
{

unsigned long b0=data[0], b1=data[1],i,k[4]={key[0],key[1],key[2],key[3]},
subkeys[16];
for(i=0;i< 16;i++){
     //Subkeys are calculated
     k[i%4]=((k[0]+k[1])+((k[2]+k[3])^(k[0]<<k[2])));
     subkeys[i]=k[i%4];
}
for(i=15; i!=-1; i--)
{
     //Process is applied in the inverse order
     b1 -= ((subkeys[i]+b0)<<9)^((subkeys[i]-b0)^((subkeys[i]+b0)>>14));
     b0 -= ((subkeys[i]+b1)<<9)^((subkeys[i]-b1)^((subkeys[i]+b1)>>14));
}
result[0]=b0;
result[1]=b1;
}
```

الشكل (10): كود فك التشفير الموافق مكتوب بلغة C

في هذه الحالة يتلقى إجراء فك التشفير النص المشفر في المعامل data ويضع النص الأصلي في المعامل result أما key فنخزن فيه المفتاح المؤلف من 128 بتاً. نحسب توابع الجولة في بداية الإجراء، ثم نطبقها بترتيب معاكس للترتيب الذي طبقت فيه عند التشفير.

لقد ضمنا إجراءي التشفير وفك التشفير باستخدام لغة البرمجة C علماً أنه يمكن وبسهولة نقل هذين الإجراءين إلى لغات برمجة أخرى.

النتائج والمناقشة:

■ قوة أسلوب التقوية المقترح

إن الضعف الأساسي في TEA وهجمات المفاتيح المتصلة والمفاتيح المتساوية التي تعرضت لها ناجمة عن جدولة المفتاح المتواضعة فيها، لذا فإن أحد الأهداف الرئيسية التي وضعناها نصب أعيننا عند تطوير أسلوب التقوية، هو جعل جدولة المفتاح في TEA أكثر قوة.

إن جدولة المفتاح التي أتينا بها بسيطة جداً أيضاً، لكنها ليست بقدر بساطة جدولة المفتاح في TEA.

إن جدولة المفتاح في TEA تتكون من إضافة الثابت $delta = (\sqrt{5}-1)*2^{31} = 9E3779B9_h$ المأخوذ من النسبة الذهبية إلى المتحول sum لذلك فإنه بمعرفة مفتاح الجولة للدورة i يصبح من السهل بمكان الحصول على المفاتيح الموافقة للدورات السابقة واللاحقة. هذه ليست الحال مع أسلوب التقوية...

إن جدولة المفتاح في أسلوبنا المقوى تسلك سلوك مولد أعداد عشوائية، ولا تعاني من الضعف الملاحظ في جدولة مفتاح TEA.

زد على ذلك أن تابع الجولة في أسلوبنا المقوى يعطي نتائج أفضل بكثير من تابع جولة TEA في الاختبارات الإحصائية، لذا يمكننا الاستنتاج بأنه أقوى.

عندما حللنا الأسلوب المقوى باستخدام الاختبار الإحصائي ENT لاحظنا بأن نتائجه عند تطبيقها على 16 دورة كانت أفضل ويمكننا مقارنتها مع نتائج TEA عند تطبيقها على 32 دورة!!!

■ **سرعة أسلوب التقوية المقترح**

قلنا إن أحد الأهداف الأساسية عند تقديم بديل للخوارزمية TEA هو أن يكون البديل بسرعة TEA. يبين الجدول (3) زمن التنفيذ مقاساً بالملي ثانية للخوارزمية TEA ولأسلوب تقويتها مع عدد الدورات الموافق:

الجدول (3): مقارنة بين أزمنة تنفيذ TEA وأسلوب تقويتها

عدد الدورات	أسلوب التقوية	TEA
8	1.29	1.07
16	***1.54***	1.24
32	1.93	***1.53***

نلاحظ أن أسلوب التقوية أبطأ من TEA من أجل دورة واحدة، وسبب ذلك أن جدولة المفتاح في أسلوب التقوية أعقد وأكبر منها في TEA لكن وبما أن أسلوب التقوية عند تطبيقه على 16 دورة يعادل في قوته TEA عند تطبيقها على 32 دورة فإن الهدف الزمني محقق، لأنه وكما نلاحظ من الجدول أن زمن تنفيذ أسلوب التقوية مع 16 دورة يعادل زمن تنفيذ TEA مع 32 دورة.

نستنتج أن سرعة أسلوب التقوية تعادل سرعة TEA، ومع ذلك يمكن اعتباره أفضل من TEA لأنه يتطلب عدد دورات أقل لتحقيق نفس مستوى الأداء.

■ **الاختبار الإحصائي ENT الذي طبقناه على أسلوب التقوية المقترح**

إن البرنامج ENT هو برنامج يطبق مجموعة من الاختبارات على سلاسل البايتات المخزنة في ملف، ثم يعطي تقريراً بنتائج تلك الاختبارات [23]. هذا البرنامج مفيد في تقييم مولدات الأعداد شبه العشوائية المستخدمة في خوارزميات التشفير وتطبيقات أخذ العينات الإحصائية وخوارزميات الضغط وغيرها من التطبيقات التي تكون فيها كثافة المعلومات في الملف مهمة.

يتم استدعاء البرنامج ent بالشكل التالي:

```
ent [ -b -c -f -t -u ] [ infile ]
```

عندها يقوم البرنامج ent بمجموعة من الاختبارات على البايتات في الملف infile (أو على البايتات المأخوذة من الدخل القياسي إذا لم يتم تحديد infile) ويعطي الخرج التالي على جهاز العرض القياسي:

```
C:\>ent test.jpg
Entropy = 7.980222 bits per byte.
Optimum compression would reduce the size
of this 329951 byte file by 0 percent.
Chi square distribution for 329951 samples is 10377.55, and randomly
would exceed this value 0.01 percent of the times.
Arithmetic mean value of data bytes is 126.8862 (127.5 = random).
```

Monte Carlo value for Pi is 3.151279300 (error 0.31 percent).
Serial correlation coefficient is -0.001115 (totally uncorrelated = 0.0).

لنشرح القيم المحسوبة أعلاه:

1) الفوضى Entropy: كثافة المعلومات المحتواة في الملف مقيسة بالبتات لكل محرف. إن قيمة الفوضى أعلاه والتي نتجت عن معالجة ملف صورة مضغوط بالتنسيق JPEG تدل على أن الملف كثيف جداً بالمعلومات، لذا من المستبعد أن ينقص الضغط من حجم الملف. وعلى العكس من ذلك فإن الفوضى لملف الكود المصدر ent.c للبرنامج ent تعادل 4.9 بت لكل محرف الأمر الذي يدل على أن الضغط المثالي لهذا الملف سينقص حجمه بنسبة 38% [24] (جرب ذلك بتشغيل البرنامج ent مستخدماً ent ent.c).

2) الاختبار Chi-square: إن الاختبار Chi-square هو الاختبار الأكثر استخداماً لعشوائية البيانات وهو حساس للغاية للأخطاء في تصميم مولدات السلاسل شبه العشوائية. يتم حساب التوزيع Chi-square للبايتات في الملف، ثم يعبر عنه برقم ثابت وبنسبة مئوية تدل على مدى قدرة سلسلة صادقة العشوائية على تجاوز الرقم الثابت المحسوب. إن هذه النسبة المئوية ما هي في الواقع إلا النسبة المئوية لكون السلسلة المختبرة غير عشوائية. فإذا كانت النسبة المئوية أكبر من 99% أو أصغر من 1% تكون السلسلة بالتأكيد غير عشوائية، أما إذا كانت النسبة بين 95% و 99% أو بين 1% و 5% فالسلسلة يشك بأنها غير عشوائية. لاحظ أن ملف JPEG، رغم كثافته الشديدة بالمعلومات بعيد جداً، عن العشوائية كما يدل اختبار Chi-square.

إن تطبيق هذا الاختبار على خرج مولدات السلاسل شبه العشوائية مثير للاهتمام. مثلاً عند تطبيق هذا الاختبار على خرج التابع ()rand العشوائي في UNIX حصلنا على:

Chi square distribution for 500000 samples is 0.01, and randomly
would exceed this value 99.99 percent of the times.

لاحظ للأسف أن مولد الأعداد العشوائية في UNIX غير عشوائي!!!

لمعلومات إضافية عن هذا الاختبار ارجع إلى [25] وللاطلاع أكثر على القوانين الرياضية له والاستفادة من آلة حاسبة تفاعلية تفضل بزيارة الارتباط [26].

3) المتوسط الرياضي Arithmetic Mean: إن المتوسط الرياضي ينتج ببساطة عن جمع كل البايتات في الملف وتقسيمها على حجم الملف. إذا كانت البيانات قريبة من العشوائية يجب أن نحصل على 127.5.

4) القيمة بي لمونتي كارلو Monte Carlo Value for Pi: في طريقة مونتي كارلو لحساب بي تُستخدم كل سلسلة متعاقبة من 6 بايتات في الملف كإحداثيات X و Y في مربع، فإذا كان بعد النقطة المولدة عشوائياً أصغر من نصف قطر الدائرة المرسومة داخل المربع فإن السلسلة المؤلفة من 6 بايتات تعتبر إصابة Hit داخل الدائرة والنسبة المئوية لهذه الإصابات يمكن استخدامها لحساب قيمة بي. إذا كانت السلسلة قريبة من العشوائية فإن بي المولدة تقترب من قيمتها الصحيحة (22/7).

5) معامل الترابط التسلسلي Serial Correlation Coefficient: يقيس معامل الترابط التسلسلي مدى اعتماد كل بايت في الملف على سابقه. إن هذا المعامل (الذي يمكن أن يكون سالباً أو موجباً) يكون قريباً من الصفر مع السلاسل العشوائية. إن ملفاً يحوي بايتات غير عشوائية كملف كود المصدر ent.c للبرنامج ent يعطي معامل ترابط تسلسلي قريباً من 0.5 (جرب ذلك بتشغيل البرنامج ent مستخدماً ent ent.c). أما البيانات التي يمكن التنبؤ بها بقوة كملفات الصور النقطية غير المضغوطة فتعطي معامل ترابط تسلسلي قريب من 1 [25].

عندما طبقنا الاختبار ENT على ملف نصي يحوي بايتات مشفرة باستخدام أسلوب التقوية، حصلنا على النتائج المشجعة الواردة في الجدول (4):

الجدول (4): نتائج تطبيق الاختبار الإحصائي ENT على أسلوب التقوية

الاختبار	النتيجة
Entropy	7,999996
Compression	0.00%
Arithmetic Mean	127,4869
Chi-square	25,00%
Monte Carlo	0,01%
Serial Correlation Coefficient	0,0000045

الاستنتاجات والتوصيات:

1) إن بساطة الخوارزمية TEA تجعل استخدامها ممكناً حتى في أصغر الدارات، وهذه ميزة يصعب الاستغناء عنها حتى ولو ثبت بالدليل القاطع أن هذه الخوارزمية عرضةٌ لتحليل الشفرة الناجح.

2) لا بد من أخذ استهلاك الذاكرة والمعالج بعين الاعتبار عند تصميم نظام تشفير فعال وعدم الاكتفاء بتقوية المشكلة الرياضية التي يستند عليها هذا النظام.

3) يمكن لمجموعة بسيطة من العمليات الرياضية أن تفضي إلى خوارزمية تشفير ناجحة وهي تغني وبحق عن الغوص عميقاً في مفاهيم نظرية الأعداد.

4) إن فضاء المفتاح يبقى موطن الضعف أو القوة الأساسي في أي نظام تشفير.

5) إن ثبات المفتاح المستخدم مع تقدم عملية التشفير قد يشكل عيباً في بعض أنظمة التشفير.

6) إن العشوائية المقترحة في أسلوب التقوية المقترح بسيطة، ويمكننا زيادة مستواها لكننا نخشى من أن يؤثر ذلك سلباً على زمن التنفيذ.

7) إن الاختبار ENT الذي استخدمناه لتقييم أسلوب التقوية فعال وشائع الاستخدام، لكنه لا يغني عن تطبيق اختبارات أخرى والقيام بالمزيد من محاولات تحليل الشفرة.

المراجع:

1. OPPLIGER, R. *"Contemporary Cryptography"*. Artech House. Norwood, MA 02062. USA. 2005, 7, 14-15.
2. *"Request for Comments Number 2828"*. Internet Security Glossary. Retrieved from: ftp://ftp.rfc-editor.org/in-notes/pdfrfc/rfc2828.txt.pdf. 2000
3. FEISTEL. H. *"Cryptography and Computer Privacy"*. Retrieved from: http://www.apprendre-en-ligne.net/crypto/bibliotheque/feistel/index.html. 1973
4. *"Tiny Encryption Algorithm"*. Wikipedia, The Free Encyclopedia. Retrieved from: http://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm.
5. BIHAM, A.; SHAMIR, A. "*Differential Cryptanalysis of the Data Encryption Standard*", Springer Verlag. 1993.
6. REDDY ANDEM, V. *"A Cryptanalysis of the Tiny Encryption Algorithm"*. Retrieved from: http://www.cs.ua.edu/SecurityResearchGroup/VRAndem.pdf. 2003
7. SUWARTADI, E.; GUNAWAN,C.; SETIJADI, P. A. ; MACHBUB, C. *"First Step Toward Internet Based Embedded Control System"*.
8. WHEELER, D.; NEEDHAM, R. *"TEA, a Tiny Encryption Algorithm"*. Retrieved from: http://www.cix.co.uk/~klockstone/tea.pdf. 1994
9. NARAYANAN, B. *"Security Distributed Systems"*. Retrieved from: http://www.cs.stevens.edu/~badri/CS649/DSSecurity.pdf. 2001
10. KARABUDAK, D. *"Security Optimization and Data Classification in Wireless Sensor Networks"*.
11. BIRYUKOV, A. ; WAGNER, D. *"Slide attacks"*. In Lars Knudsen, editor, Fast software encryption: 6th International Workshop, FSE'99, Rome, Italy, March 24-26, (1999): proceedings, volume 1636 of Lecture Notes in Computer Science, pages 245-259, Berlin, Germany / Heidelberg, Germany / London, UK / etc., (1999). Springer-Verlag. ISBN 3-540-66226-X (softcover). Retrieved from: http://www.cs.berkeley.edu/~daw/papers/slide-fse99.ps.
12. FLEMING, R. *"An attack on a weakened version of TEA"*. sci.crypt posting, (October 1996). Retrieved from: http://groups.google.com/group/sci.crypt/msg/1e7f50e480e3c3b0.
13. KELSEY, J.; SCHNEIER, B.; WAGNER, D. *"Key-schedule cryptanalysis of IDEA"*, G-DES, GOST, SAFER, and Triple-DES. Lecture Notes in Computer Science, 1109: 237-251, 1996. ISSN 0302-9743. Retrieved from: http://www.schneier.com/paper-key-schedule.pdf.
14. *"New Xbox security cracked by Linux fans"*. Available at: http://news.zdnet.co.uk/software/developer/0,39020387,2123851,00.htm.
15. KELSEY, J.; SCHNEIER, B. ; WAGNER, D. *"Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X NewDES, RC2, and TEA"*. Lecture Notes in Computer Science, 1334: 233-246, 1997. ISSN 0302-9743. Retrieved from: http://www.schneier.com/paper-relatedkey.pdf.
16. HERNANDEZ, J. C.; SIERRA, J. M.; RIBAGORDA, A.; RAMOS, B.; MEX-PERERA, J. C. *"Distinguishing TEA from a random permutation: Reduced round versions of TEA do not have the SAC or do not generate random numbers"*. In Proceedings of the IMA Int. Conf. on Cryptography and Coding, 2001,374-377.

17. HERNANDEZ, J. C.; SIERRA, J. M.; ISASI, P.; RIBAGORDA, A. *"Genetic cryptanalysis of two rounds TEA"*. Lecture Notes in Computer Science, 2331: 1024-1031, 2002. ISSN 0302-9743.
18. HERNANDEZ, J. C.; ISASI, P.; RIBAGORDA, A. *"An application of genetic algorithms to the cryptanalysis of one round tea"*. In Proceedings of the 2002 Symposium on Artificial Intelligence and its Application, 2002.
19. HERNANDEZ, J. C.; SIERRA, J. M.; ISASI, P.; RIBAGORDA,A. *"Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher TEA"*. In Proceedings of the 2003 Congress on Evolutionary Computation, 2003.
20. MOON, D.; HWANG, K.; LEE, W.; LEE, S.; LIM, J. *"Impossible differential cryptanalysis of reduced round XTEA and TEA"*. Lecture Notes in Computer Science, 2365: 2002,49-60. ISSN 0302-9743.
21. HONG, D.; KO, Y.; CHANG, D.; LEE, W.; LIM, J. *"Differential cryptanalysis of XTEA"*. Technical Report TR03_13, Center for the Information Security and Technologies (CIST), Seoul, Korea, 2003a.
22. HONG, S.; HONG, D.; KO, Y.; CHANG, D.; LEE, W.; LEE, S. *"Differential cryptanalysis of TEA and XTEA"*. In Proceedings of ICISC ,2003, 2003b.
23. *"A Pseudorandom Number Sequence Test Program"*. Available at: http://www.fourmilab.ch/random/.
24. HAMMING, R. W. *"Coding and Information Theory"*. Englewood Cliffs NJ: Prentice-Hall, 1980.
25. KNUTH, D. E. *"The Art of Computer Programming, Volume 2 / Seminumerical Algorithms"*. Reading MA: Addison-Wesley, 1969. ISBN 0-201-89684-2.
26. *"Chi-Square Calculator"*. Available at: http://www.fourmilab.ch/rpkp/experiments/analysis/chiCalc.html.