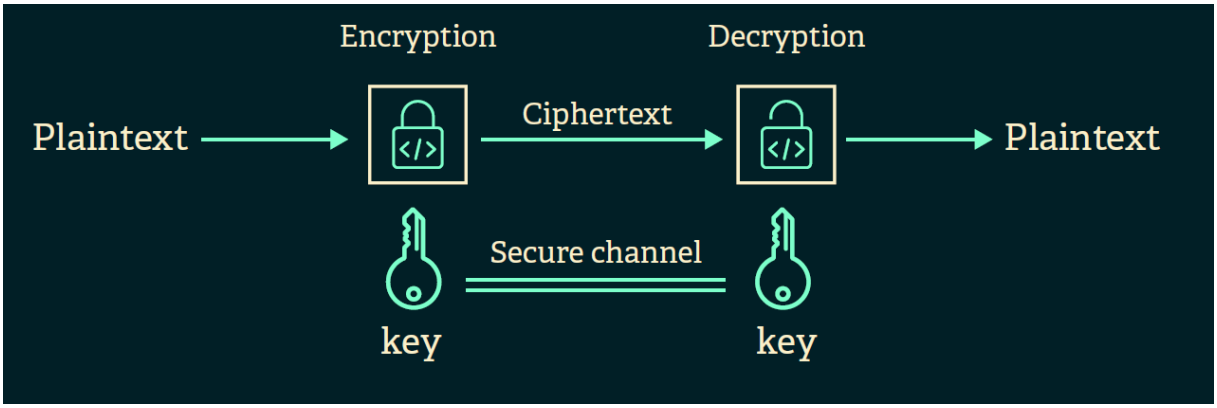
TEA(tint encryption algorithm)

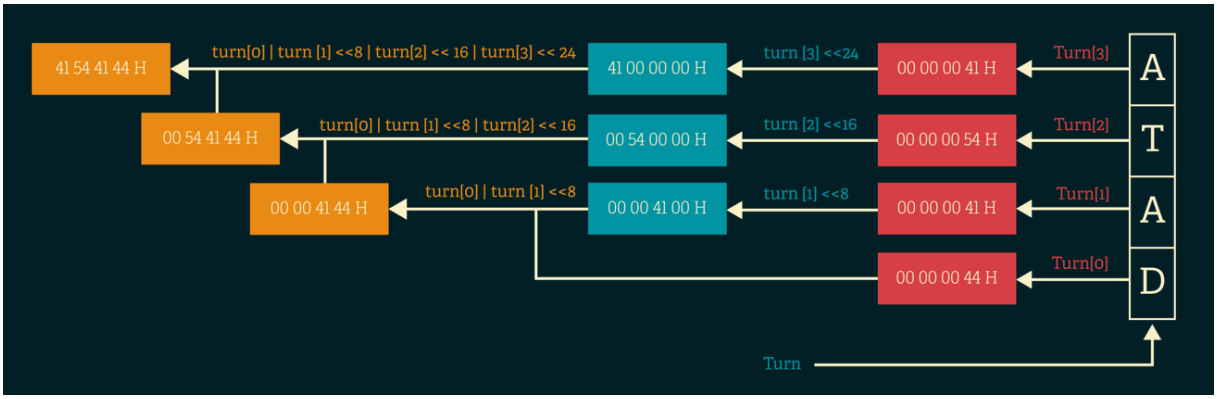


Projemizde iletilen veriler, kullanılarak şifrelenecek ve şifresi çözülecek Sınırlı kaynağa sahip olan ancak yine de hız ve alanla ilgili endişeleri olan mobil sistemler için özel olarak tasarlanmış en verimli ve en hızlı yazılım şifreleme algoritmalarından biri Tiny Encryption Algorithm'dir (uygun kısaltması TEA ile bilinir).Tea'yi C++'da uygulayın: Şifrenin işlevselliğini basitleştirmek için yüksek seviyeli dilde [64-bit] giriş dizesini almak için çay şifresini uyguladık. Bu yüzden şu beş işlevi uyguladık:şifrelemeşifre çözme Birleştir ve Böl İki yardımcı işlev.TEA Ardından bu kodu birden çok 64 bit blokta çalışacak şekilde yükseltin.

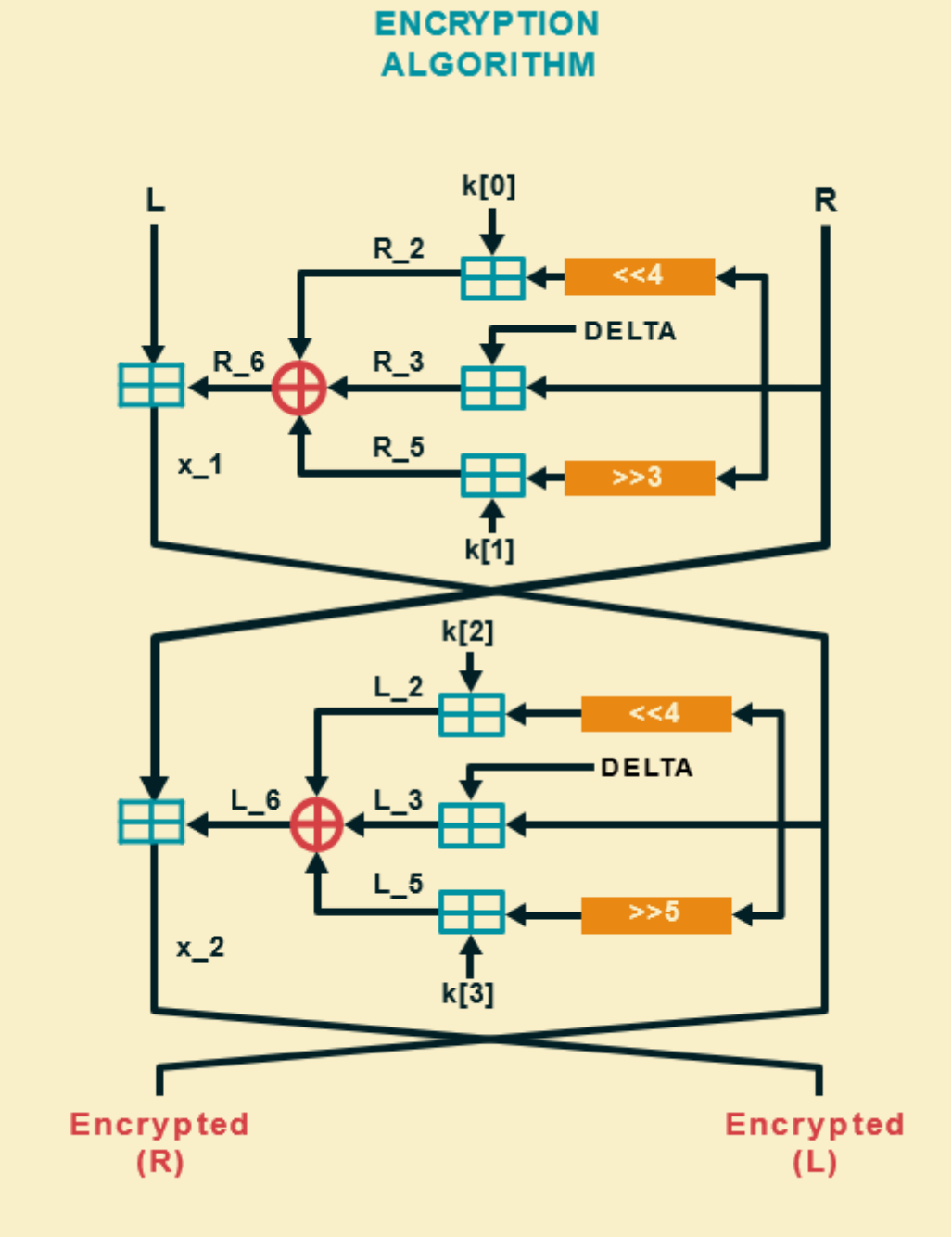
TEA implementasyon

COMBİNE:

1. **Bu işlev, her dört karakteri bir tam sayıda birleştirir.**
2. **Giriş 8 karakterdir, bu nedenle iki tam sayı dizisine ihtiyacımız var**

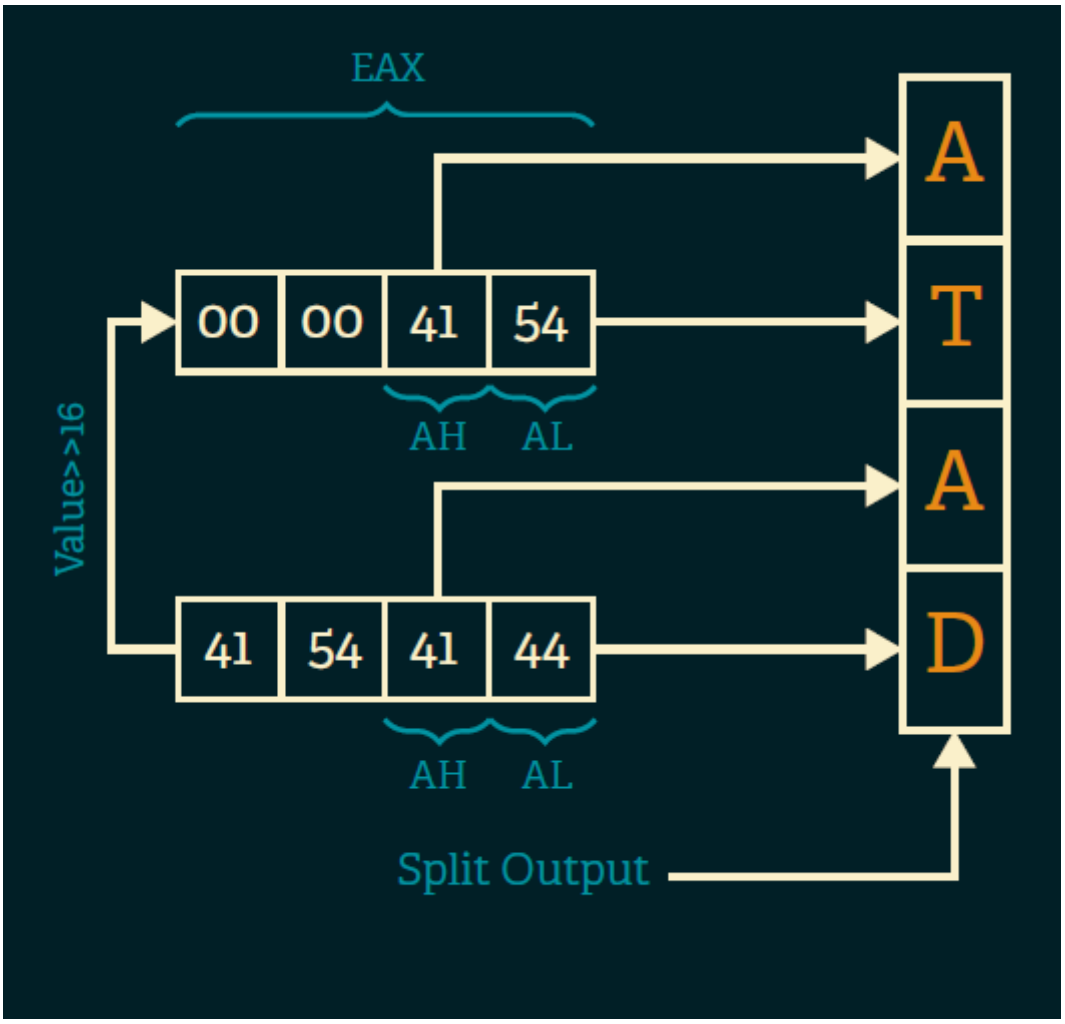


ENCRYPTİON ALGORİTM

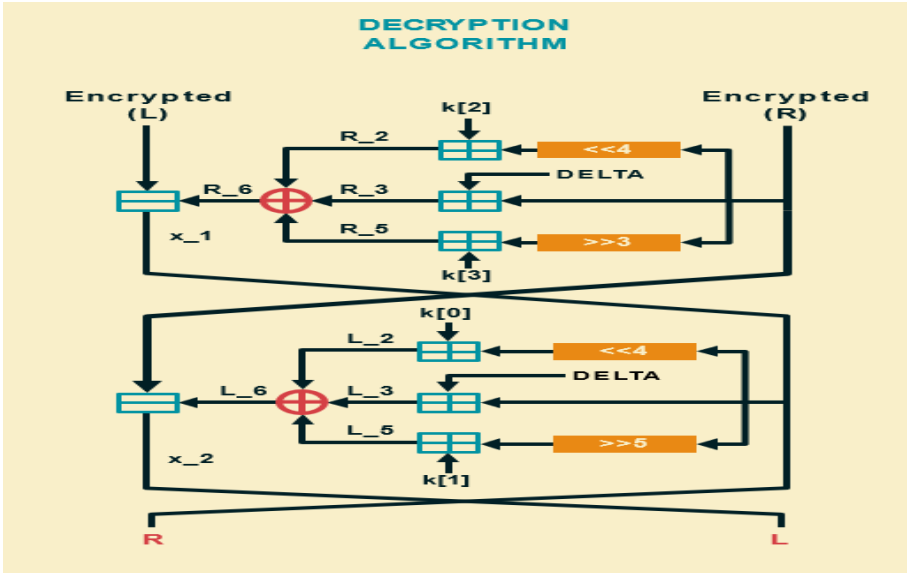


SPLİT

1. Bu işlev, birleştirme işleminin tersidir.
2. Şifreleme veya şifre çözme işleminden döndürülen diziyi kullanır, her 32 bitlik tamsayı 4 karaktere bölünmüştür.



DECRYPTİON ALGORİTHM

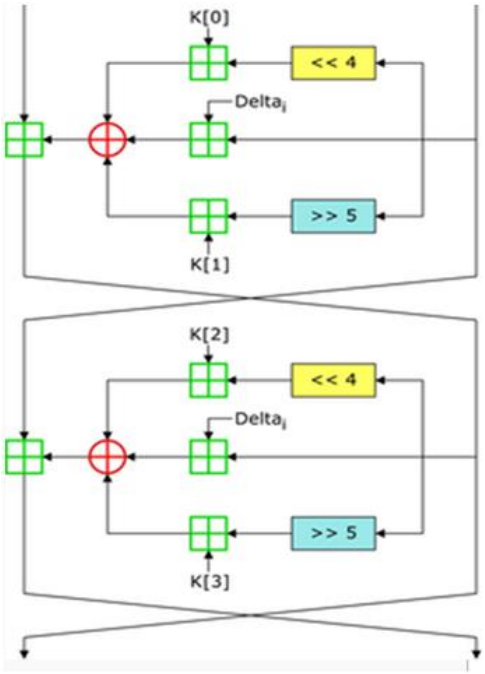


Gelişimi ve Temel Özellikleri

- Tiny Encryption Algorithm(TEA), Cambridge Bilgisayar Laboratuar'ında David Wheeler ve Roger Needham tarafından geliştirilmiştir.
- İlk olarak 1994 yılında Leuven'de Fast Software Algorithm (Hızlı Yazılım Şifreleme) atölyesinde sunulmuştur. Tiny şifreleme algoritması blok şifrelemeyi kullanır.
- TEA, "XOR, ADD ve SHIFT" gruplarını içeren karışık cebirsel işlemleri ve Fiestel ağını kullanan şifrelemedir.
- Basitliği ve çoğu şifreleme algoritmalarından daha kısa satırdan oluşan uygulamasıyla dikkate değerdir.
- Çok kısa olan kod boyutu ve basit algoritması sayesinde özellikle kod boyutunun oldukça sınırlı olduğu gömülü sistemlerde oldukça popüler olan bir şifreleme algoritmasıdır.
- TEA var olan en hızlı ve en etkili algoritmalardan birisidir.
- TEA, hafızada kaplanan yeri minimize etmek ve hızı maksimize etmek için geliştirilmiş bir algoritmadır.
- 64 bitlik bloklar kullanır. Bu 64 veri bloğunu 128 bitlik anahtar ile şifreler.
- 128 bitlik K anahtarı 32 bitlik bloklara bölünür. K = (K[0], K[1], K[2], K[3])
- Değişebilmesine rağmen 64 adet Fiestel turu – 32 döngü tavsiye edilmektedir. (2 Fiestel turu = 1 döngü) Terminolojide 2'li yani çift tura Cycle = Döngü denmektedir.
- TEA, Shannon'un önerdiği ve güvenli bir blok şifreleme için gerekli olan ***karıştırma*** *(confusion)* ve ***yayılma*** *(diffusion)* özelliklerini sağlayan önemli bir şifreleme yöntemidir. Karıştırma(confusion) şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma(diffusion) açık metindeki izlerin şifreli metinde sezilmemesini sağlamak için kullanılır. (Blok şifreler, Shannon'un önerdiği karıştırma ve yayılma tekniklerine dayanır.)
- Tam bir yayılma sağlar. Plain text, metin'de yapılan tek bir bit-in değişikliği cipher text, şifreli mesajda 32 bitlik değişikliğe sebep olur. Modern bir bilgisayardaki ya da çalışma alanındaki performansı etkileyicidir.

TEA Tur(Round) Yapısı

TEA için “2 Fiestel turu – 1 çevrim” yapısı aşağıdaki şekilde gösterilmiştir.



Şekil 3: İki Fiestel Turu 1

TEA Güvenlik Durumu

- TEA en güvenli algoritmalardan biridir.
- Massey ve Xuejia Lai tarafından tasarlanan IDEA algoritması kadar güvenli olduğu söylenebilir.
- IDEA’ da kullanılan aynı karışık cebirsel grupları kullanmasına rağmen daha basit ve de daha hızlıdır. Ayrıca TEA hiçbir kuruma ait olmamasına rağmen IDEA, İsviçre’ de bulunan Ascom-Tech AG tarafından patentlenmiştir.
- TEA algoritmasının birkaç zayıf noktası vardır. Bunlardan en dikkat çekeni ise “equivalent key” durumudur. Bir şifreleme sisteminde, bir metnin K ve K* anahtarlarıyla şifrelenmesi sonucu aynı şifreli mesaj oluşuyorsa bu anahtarlara “equivalent key” denir. İyi dizayn edilmiş bir şifrelemede, equivalance – eşitlik sınıfına dahil olan her anahtarın plain text → cipher text dönüşümünü gösteren kesin bir haritası olması ve eşit, denk anahtarların bulunmaması gerekir. Böyle bir durumda 2^k eşitlik sınıfı bulunması beklenir. TEA yönteminde 128 bitlik anahtarlar vardır. Bu durumda 2^128 eşitlik sınıfı bulunması beklenirken araştırmalar 2^126 sınıf olduğunu göstermiştir. Bu zayıflığın sebebi ise tur fonksiyonudur. Yani etkili anahtar boyutu 126 bit olmaktadır. Yani aslında TEA 128 bitlik anahtarlar kullansa da 126 bitin sağlayacağı

güvenliği sağlamaktadır. Bu durum TEA şifrelemenin, blok şifrelemeye dayalı hash fonksiyon içinde kullanılmasını uygunsuz kılmaktadır.

- TEA ayrıca ilişkili anahtar saldırılarına karşı duyarlıdır. Bu zafiyetinden dolayı da TEA algoritmasının birkaç uyarlaması tasarlandı.

TEA Çalışma Yapısı

Şifreleme Yapısı

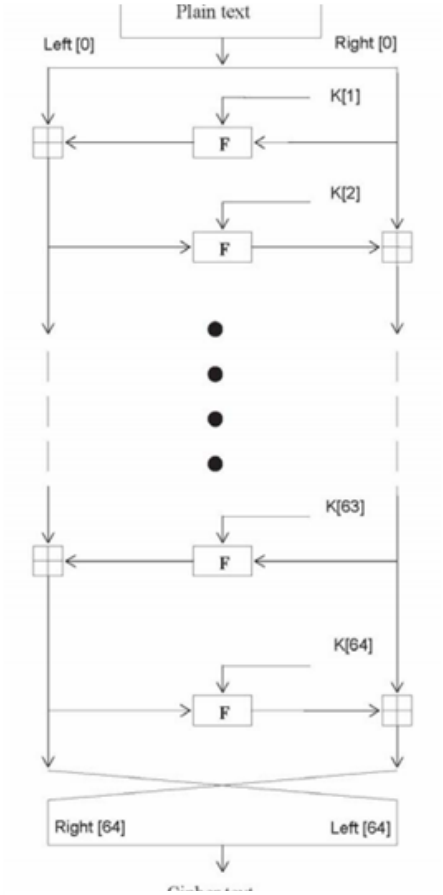
Wheeler, 1994'de Cambridge Üniversitesi bilgisayar laboratuarında TEA algoritmasının şifreleme şablonunu, programını geliştirmiştir. Aşağıda C dilindeki şifreleme yapısı gösterilmiştir. Burada veri V[0] – V[1]'de depolanmıştır. Anahtar algoritması basittir ve 128 bitlik K anahtarı 32 bitlik 4 bloğa bölünmüştür K={k[0], k[1], k[2], k[3]}.

Bu algoritma, yazılımda DES algoritmasının yerine kullanılabilir ve neredeyse her programa yazılabilecek kadar kısadır. 64 turla – 32 çevrimle hız kuvvetli bir hedef olmamasına rağmen TEA, 16 tur içeren iyi bir DES yazılım uygulamasından 3 kat daha hızlıdır.

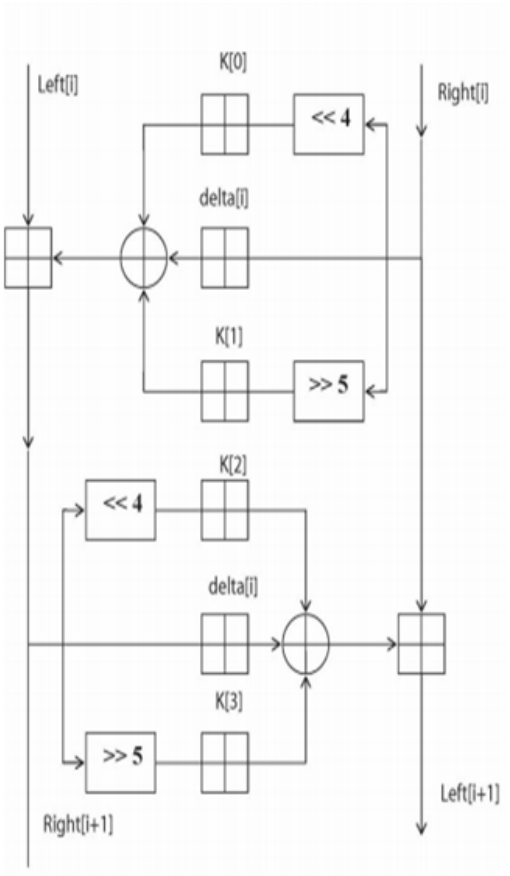
Şifreleme algoritmasının girdileri plain text bloğu ve anahtar K' dır. Plain text P = (Left[0], Right[0]) ve cipher text C = (Left[64], Right[64]). Metin 2'ye bölünmüştür, Left[0], Right[0]. Her yarım, 64 işlem turunda diğer yarımı şifrelemek için kullanılır ve 64 tur sonunda şifreli mesajı üretmek için bu 2 yarım birleştirilir.

- Her i turunda, bir önceki turdan gelen Left[i-1] ve Right[i-1] girdisi vardır.
- Her alt(sub) K anahtarı birbirinden farklıdır.
- Sabit delta değeri, kriptografik olarak anahtarların birbirleriyle bir ilişkilerinin olmaması için altın sayı oranından (golden number ratio) oluşturulmuştur.

$$delta = (\sqrt{5} - 1) * 2^{31} = 9E3779B9_h$$



Şekil 4: TEA Şifreleme Yapısı 1



Şekil 5: TEA Şifrelemede 1 Çevrim 1

$$Left\ [i+1] \ = \ Left[i] \boxplus F\ (\ Right[i], K\ [0, 1], delta[i]\ )$$

$$F(M, K[j,k], delta[i]\ ) = ((M << 4) \boxplus K[j]) \oplus (M \boxplus delta[i]\ ) \oplus ((M >> 5) \boxplus K[k])$$

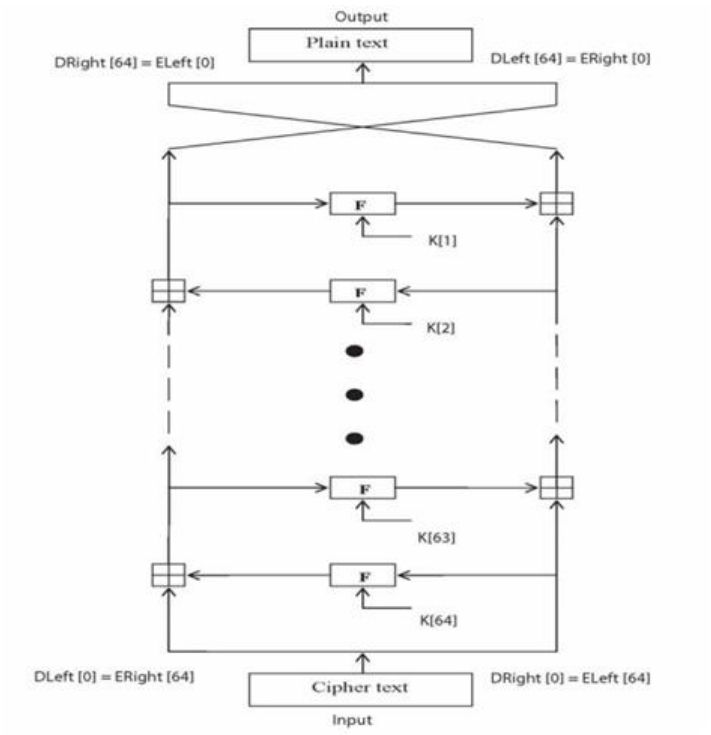
y = V[0] yani metnin sol yarımıdır.

KODDA → y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1];

Yani 3 adet toplam işleminin sonucu XOR işlemine girmektedir. sum terimi her iterasyonda deltanın değeri kadar arttırılmaktadır.

Şifre Çözme Yapısı

Kod çözme aslında temel olarak şifreleme süreci ile aynıdır. Decode sırasında şifreli mesaj, cipher text algoritmanın girdisi olarak kullanılır ama alt anahtarlar(sub keys) K[i] ters sırada kullanılır.



Şekil 6: TEA Şifre Çözme Yapısı 1

Kullanılan Terimler

- "<<" terimi sola kaydırma anlamındadır. Örneğin, x<<4 demek x' i 4 bit sola kaydır demektir.
- ">>" terimi sağa kaydırma anlamındadır. Örneğin, x>>4 demek x' i 4 bit sağa kaydır demektir.
- "^" terimi XOR işlemini ifade etmektedir.
- " ⊞ " ise toplama işlemini ifade etmektedir.
- " ⊕ " ise XOR işlemi için kullanılmaktadır.

Referanslar

http://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm

http://edipermadi.files.wordpress.com/2008/06/tea-spec.pdf

http://www.codeproject.com/KB/mobile/teaencryption.aspx

http://www.udea.com.tr/1/UN-0506v01_Sifreleme.pdf

http://wapedia.mobi/en/Tiny_Encryption_Algorithm#1.

http://cs.ua.edu/SecurityResearchGroup/VRAndem.pdf