Tiny Encryption Algorithm

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

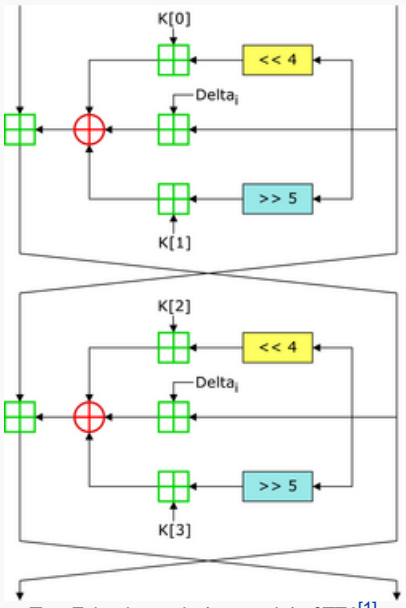
Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Español
Français
Hrvatski
Italiano
Polski
Русский
中文
Edit links

In cryptography, the **Tiny Encryption Algorithm** (**TEA**) is a block cipher notable for its simplicity of description and implementation, typically a few lines of code. It was designed by David Wheeler and Roger Needham of the Cambridge Computer Laboratory; it was first presented at the Fast Software Encryption workshop in Leuven in 1994, and first published in the proceedings of that workshop.[4]

The cipher is not subject to any patents.

TEA



Two Feistel rounds (one cycle) of TEA[1]

General

Designers	Roger Needham, David Wheeler
First published	1994
Successors	XTEA

Cipher detail

Key sizes	128 bits
Block sizes	64 bits
Structure	Feistel network
Rounds	variable; recommended 64 Feistel rounds (32 cycles)

Best public cryptanalysis

TEA suffers from equivalent keys (Kelsey et al., 1996) and can be broken using a related-key attack requiring 2^{23} chosen plaintexts and a time complexity of 2^{32}.[2] The best structural cryptanalysis of TEA in the standard single secret key setting is the zero-correlation cryptanalysis breaking 21 rounds in $2^{121.5}$ time with less than the full code book.[3]

Properties [edit]

TEA operates on two 32-bit unsigned integers (could be derived from a 64-bit data block) and uses a 128-bit key. It has a Feistel structure with a suggested 64 rounds, typically implemented in pairs termed *cycles*. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle. Different multiples of a magic constant are used to prevent simple attacks based on the symmetry of the rounds. The magic constant, 2654435769 or $9E3779B9_{16}$ is chosen to be $\lfloor 2^{32}/\phi \rfloor$, where ϕ is the golden ratio.[4]

TEA has a few weaknesses. Most notably, it suffers from equivalent keys—each key is equivalent to three others, which means that the effective key size is only 126 bits.[5] As a result, TEA is especially bad as a cryptographic hash function. This weakness led to a method for hacking Microsoft's Xbox game console, where the cipher was used as a hash function.[6] TEA is also susceptible to a related-key attack which requires 2^{23} chosen plaintexts under a related-key pair, with 2^{32} time complexity.[2] Because of these weaknesses, the XTEA cipher was designed.

Versions [edit]

The first published version of TEA was supplemented by a second version that incorporated extensions to make it more secure. *Block TEA* (sometimes referred to as XTEA) operates on arbitrary-size blocks in place of the 64-bit blocks of the original.

A third version (XXTEA), published in 1998, described further improvements for enhancing the security of the Block TEA algorithm.

Reference code [edit]

Following is an adaptation of the reference encryption and decryption routines in C, released into the public domain by David Wheeler and Roger Needham:[4]

```c
#include <stdint.h>

void encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i < 32; i++) {                       /* 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;
}

void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* 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;
}
```

Note that the reference implementation acts on multi-byte numeric values. The original paper does not specify how to derive the numbers it acts on from binary or other content.

See also [edit]

- RC4 - A stream cipher that, just like TEA, is designed to be very simple to implement.
- XTEA - First version of Block TEA's successor.
- XXTEA - Corrected Block TEA's successor.
- Treyfer - A simple and compact encryption algorithm with 64 bit key size and block size.

Notes [edit]

1. ^ Matthew D. Russell (27 Feb 2004). "Tinyness: An Overview of TEA and Related Ciphers". Archived from the original on 12 August 2007.
2. ^ [a] [b] Kelsey, John; Schneier, Bruce; Wagner, David (1997). "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X NewDES, RC2, and TEA". *Lecture Notes in Computer Science* **1334**: 233–246. doi:10.1007/BFb0028479.
3. ^ Bogdanov, Andrey; Wang, Meiqin (2012). "Zero-Correlation Linear Cryptanalysis with Reduced Data Complexity" (PDF). *Lecture Notes in Computer Science* (Fast Software Encryption 2012) **7549**: 29–48. doi:10.1007/978-3-642-34047-5_3.
4. ^ [a] [b] [c] Wheeler, David J.; Needham, Roger M. (1994-12-16). "TEA, a tiny encryption algorithm". *Lecture Notes in Computer Science* (Leuven, Belgium: Fast Software Encryption: Second International Workshop) **1008**: 363–366. doi:10.1007/3-540-60590-8_29.
5. ^ Kelsey, John; Schneier, Bruce; Wagner, David (1996). "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES" (PDF). *Lecture Notes in Computer Science* **1109**: 237–251. doi:10.1007/3-540-68697-5_19.
6. ^ Michael Steil. "17 Mistakes Microsoft Made in the Xbox Security System". Archived from the original on 16 April 2009.

References [edit]

- Andem, Vikram Reddy (2003). "A Cryptanalysis of the Tiny Encryption Algorithm, Masters thesis" (PDF). Tuscaloosa: The University of Alabama.
- Hernández, Julio César; Isasi, Pedro; Ribagorda, Arturo (2002). "An application of genetic algorithms to the cryptoanalysis of one round TEA". *Proceedings of the 2002 Symposium on Artificial Intelligence and its Application*.
- Hernández, Julio César; Sierra, José María; Isasi, Pedro; Ribargorda. Arturo (2003). "Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher TEA". *Proceedings of the 2003 Congress on Evolutionary Computation* **3**: 2189. doi:10.1109/CEC.2003.1299943.

- Hernández, Julio César; Sierra, José María; Ribagorda, Arturo; Ramos, Benjamín; Mex-Perera, J. C. (2001). "Distinguishing TEA from a random permutation: Reduced round versions of TEA do not have the SAC or do not generate random numbers" (PDF). *Proceedings of the IMA Int. Conf. on Cryptography and Coding 2001*: 374–377. doi:10.1007/3-540-45325-3_34.
- Moon, Dukjae; Hwang, Kyungdeok; Lee, Wonil; Lee, Sangjin; Lim, Jongin (2002). "Impossible differential cryptanalysis of reduced round XTEA and TEA" (PDF). *Lecture Notes in Computer Science* **2365**: 49–60. doi:10.1007/3-540-45661-9_4.
- Hong, Seokhie; Hong, Deukjo; Ko, Youngdai; Chang, Donghoon; Lee, Wonil; Lee, Sangjin (2003). "Differential cryptanalysis of TEA and XTEA". *In Proceedings of ICISC 2003*. doi:10.1007/978-3-540-24691-6_30.

External links [edit]

- Test vectors for TEA
- JavaScript implementation of XXTEA with Base64
- PHP implementation of XTEA (German language)
- JavaScript implementation of TEA
- JavaScript and PHP implementations of XTEA (Dutch text)
- LGPL Java/J2ME implementation of TEA
- AVR ASM implementation
- Common Lisp implementation of TEA

v · t · e	**Block ciphers (security summary)**
Common algorithms	AES · Blowfish · DES (Internal Mechanics, Triple DES) · Serpent · Twofish
Less common algorithms	Camellia · CAST-128 · IDEA · RC2 · RC5 · SEED · ARIA · Skipjack · **TEA** · XTEA
Other algorithms	3-Way · Akelarre · Anubis · BaseKing · BassOmatic · BATON · BEAR and LION · CAST-256 · Chiasmus · CIKS-1 · CIPHERUNICORN-A · CIPHERUNICORN-E · CLEFIA · CMEA · Cobra · COCONUT98 · Crab · Cryptomeria/C2 · CRYPTON · CS-Cipher · DEAL · DES-X · DFC · E2 · FEAL · FEA-M · FROG · G-DES · GOST · Grand Cru · Hasty Pudding cipher · Hierocrypt · ICE · IDEA NXT · Intel Cascade Cipher · Iraqi · KASUMI · KeeLoq · KHAZAD · Khufu and Khafre · KN-Cipher · Ladder-DES · Libelle · LOKI (97, 89/91) · Lucifer · M6 · M8 · MacGuffin · Madryga · MAGENTA · MARS · Mercy · MESH · MISTY1 · MMB · MULTI2 · MultiSwap · New Data Seal · NewDES · Nimbus · NOEKEON · NUSH · PRESENT · Q · RC6 · REDOC · Red Pike · S-1 · SAFER · SAVILLE · SC2000 · SHACAL · SHARK · Simon · SMS4 · Speck · Spectr-H64 · Square · SXAL/MBAL · Threefish · Treyfer · UES · Xenon · xmx · XXTEA · Zodiac
Design	Feistel network · Key schedule · Lai-Massey scheme · Product cipher · S-box · P-box · SPN · Avalanche effect · Block size · Key size · Key whitening (Whitening transformation)
Attack (cryptanalysis)	Brute-force (EFF DES cracker) · MITM (Biclique attack, 3-subset MITM attack) · Linear (Piling-up lemma) · Differential (Impossible · Truncated · Higher-order) · Differential-linear · Integral/Square · Boomerang · Mod *n* · Related-key · Slide · Rotational · Timing · XSL · Interpolation · Partitioning · Davies' · Rebound · Weak key · Tau · Chi-square · Time/memory/data tradeoff
Standardization	AES process · CRYPTREC · NESSIE
Utilization	Initialization vector · Mode of operation · Padding
v · t · e	**Cryptography**
	History of cryptography · Cryptanalysis · Cryptography portal · Outline of cryptography
	Symmetric-key algorithm · Block cipher · Stream cipher · Public-key cryptography · Cryptographic hash function · Message authentication code · Random numbers · Steganography

Categories: Block ciphers | Broken block ciphers | Feistel ciphers | Free ciphers | University of Cambridge Computer Laboratory