International Journal of Inventive Engineering and Sciences (IJIES)
ISSN: 2319-9598 (Online), Volume-13 Issue-11, November 2026

Yuga Cipher v2: An Open Cryptanalysis Challenge for a 256-Bit ARX–S-Box Block Cipher with Yuga-Derived Parameters

S. Ravichandran

Abstract: ***This paper presents Yuga Cipher v2, a 128-bit block cipher used in counter (CTR) mode with a 256-bit secret key, together with an open, publicly verifiable cryptanalysis challenge. Following Kerckhoffs's principle, we disclose the full algorithm, reference source code, all ciphertexts, nonces, and known plaintext–ciphertext pairs; we withhold only keys and secret messages. The design draws its constants from the traditional Indian reckoning of the yugas: each of the 1,536 subkeys, derived from the master key via SHAKE-256, is a number of seconds less than one yuga (4,20,000 years), and each round constant spans four yugas (17,20,000 years). Each of the 256 cycles combines an 8-bit S-box layer with add–rotate–XOR mixing under six subkeys. Empirical avalanche measurements show full diffusion after two cycles. The challenge has five levels, from a deliberately weak 4-cycle version to the full 256-cycle Cipher, each with an independent key, so results on reduced versions show how security grows with the number of cycles. SHA-256 commitments to every key and secret message let anyone verify any claimed solution and prevent keys from changing after publication. The security claim for the full cipher is stated as a hypothesis for community testing, not as a proof.***

Keywords: ***Block Cipher, Symmetric Cryptography, Cryptanalysis Challenge, ARX, S-Box, Key Schedule, CTR Mode, Kerckhoffs's Principle, Key Commitment, Yuga***

I. INTRODUCTION

Trust in a symmetric cipher is earned through sustained public cryptanalysis rather than through the size of its key space alone. The Advanced Encryption Standard was selected after an open, multi-year competition [1], and public challenges with verifiable targets, such as the RSA Factoring Challenge, have long been used to measure practical attack capability [2]. A new design benefits from the same openness: the algorithm is fully specified, concrete targets are published, and anyone may attempt to break them.

This paper makes three contributions:

- **Design:** Yuga cipher v2, a 256-cycle ARX–S-box block cipher whose subkey ranges and round constants are drawn from the traditional Yuga time scale. Its security also rests on a uniformly random 256-bit master key.
- **Evaluation:** Empirical plaintext and key avalanche measurements across cycle counts from 1 to 256.
- **Verifiable Challenge:** A five-level challenge with known plaintext–ciphertext pairs and SHA-256 commitments, enabling objective verification of any break and measurement of the security margin as a function of cycle count.

Section 2 reviews related work. Section 3 describes the yuga time scale, and Section 4 specifies the algorithm. Section 5 reports the empirical evaluation, Section 6 states the security claim, Section 7 describes the challenge, and Section 8 discusses limitations. The Appendices provide the complete challenge data and reference implementation.

II. RELATED WORK

Substitution–Permutation Designs: AES [1] combines an 8-bit S-box with linear mixing; its S-box, used unchanged in this work, has well-understood differential uniformity and nonlinearity.

ARX Designs: Salsa20/ChaCha [3] and the SPECK family [4] use only modular addition, rotation, and XOR, enabling efficient software implementations; their analysis relies on the nonlinearity of modular addition. Yuga Cipher v2 combines both approaches, in the spirit of hybrid designs that interleave S-box and ARX layers.

Generic Attacks: Differential [5] and linear [6] cryptanalysis are the primary tools for evaluating round-reduced versions; slide attacks [7] target ciphers with self-similar round functions, which the per-cycle round constants and distinct subkeys are intended to prevent.

Primitives and Mode: The key schedule uses the SHAKE-256 extendable-output function [8]; CTR mode follows NIST SP 800-38A [9].

Design Principle: The evaluation follows Kerckhoffs's principle: security must depend only on key secrecy.

III. THE YUGA TIME SCALE

One Julian year is taken as 365.25 days = 31,557,600 seconds. Two spans are used:

Quantity	Years	Seconds (Exact Integer)	Approx. Value
ONE_YUGA_SECONDS	4,20,000	13,254,192,000,000	$2^{43.6}$
FOUR_YUGA_SECONDS	17,20,000	54,279,072,000,000	$2^{45.6}$

These spans define the range of each subkey and the round constants; they are not the secret key space. A secret drawn only from a range of about 2^{46} values could be searched exhaustively with modest hardware. The secret is therefore a uniformly random 256-bit master key, from

Manuscript received on 30 September 2026 | First Revised Manuscript received on 05 November 2026 | Second Revised Manuscript received on 10 November 2026 | Manuscript Accepted on 15 November 2026 | Manuscript published on 30 November 2026.

*Correspondence Author(s)
S. Ravichandran*, Independent Researcher, Puttur (Andhra Pradesh), India. Email ID: saktheee2009@gmail.com, ORCID ID: 0009-0003-6094-2534

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open-access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

which all subkeys are derived, giving an effective key space of 2^{256}. The yuga constants play the role of "nothing-up-my-sleeve" numbers: they are publicly motivated and leave no room for hidden structure.

IV. ALGORITHM SPECIFICATION

Item	Value
Master key K	256 bits, uniformly random
Block size	128 bits, two 64-bit words A, B (big-endian)
Cycles	256 (full); N in a reduced version (first N cycles)
Subkeys	6 per cycle; 1,536 in the full cipher; each in [0, ONE_YUGA_SECONDS)
Non-linear layer	AES S-box applied to each byte
Mixing	Addition mod 2^{64}, rotation, XOR
Mode	CTR: C = P XOR E_K (nonce + i), 128-bit public nonce

A. Key Schedule

SHAKE-256 is initialised with "YUGA-CIPHER-v2|" || K. Read its output in 8-byte big-endian words; mask each word to its low 44 bits, and accept it only if it is smaller than ONE_YUGA_SECONDS (rejection sampling, yielding a uniform value in the yuga range). Accepted values, in order, form subkeys $k_{r,0}$ ... $k_{r,5}$ for cycles r = 0, 1, 2, ...

B. Round Constant

c = ((r+1) × FOUR_YUGA_SECONDS) mod 2^{64}; C_r = c ⊕ rotl(c,23) ⊕ rotl(c,47).

C. One Cycle

A = (A + k0) mod 2^64; B = B xor k1
A = S(A); B = S(B) # S-box on each byte
A = (A + rotl(B,13)) mod 2^64
B = rotl (B xor A, 29)
A = A xor k2; B = (B + k3) mod 2^64
A = S(A)
B = B xor rotl(A,41)
A = (A + B) mod 2^64
A = A xor k4 xor C_r ; B = B xor k5
(A, B) = (B, A) # swap halves

The output block is A || B. Byte i of a word occupies bits 8i..8i+7 from the least significant end. The reference implementation (Appendix B) is normative.

D. Test Vector

Field	Value
Key	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
Nonce	00000000000000000000000000000000
Plaintext	"Yuga Cipher v2 test vector." (ASCII, 27 bytes)
Ciphertext	e1ab493643561e4b93dfc7dddb659a4c58e5c7b78a73c2e135e313

V. EMPIRICAL EVALUATION

For each cycle count, we drew random keys and blocks, flipped one randomly chosen bit of the input block (plaintext avalanche) or the 256-bit master key (key avalanche), and recorded the fraction of the 128 output bits that changed. An ideal cipher gives 0.5.

Cycles	Samples	Plaintext Avalanche	Min. Bits Flipped	Key Avalanche
1	300	0.2748	8 / 128	0.4988
2	300	0.4989	46 / 128	0.5020
3	300	0.4973	47 / 128	0.5038
4	300	0.4979	49 / 128	0.5012
8	300	0.5033	47 / 128	0.4962
16	300	0.5003	47 / 128	0.5010
64	100	0.5045	52 / 128	0.4945
256	100	0.4965	51 / 128	0.5000

A single cycle is clearly insufficient (0.27 average, as few as 8 bits changed). From two cycles onward, both averages are within statistical noise of 0.5 and the minimum stays well above zero. The key avalanche is near 0.5 even for one cycle, as expected, because every subkey is an output of SHAKE-256. The full cipher therefore uses more than 100 times the number of cycles needed for full diffusion. Avalanche is a necessary but not sufficient condition for security; differential and linear trail bounds are left to the challenge (Section 7) and future work.

VI. SECURITY CLAIM

For the full 256-cycle cipher, recovering the key, or any unknown plaintext, from the data in Appendix A is claimed to require work comparable to exhaustive search of a 256-bit key. The subkeys add no key space beyond the master key. This claim is a hypothesis to be tested, not a proof. Levels with 4, 8 and 16 cycles are intentionally weak and may be breakable; each level uses an independent random key, so breaking one level reveals nothing about another.

VII. THE OPEN CHALLENGE

Level	Cycles	Subkeys	Secret Message	Known Pairs
1	4	24	34 bytes	4
2	8	48	34 bytes	4
3	16	96	35 bytes	4
4	64	384	35 bytes	4
5	256	1536	50 bytes	4

- **Solution:** (a) the exact secret message of any level; (b) the master key of any level; or (c) a documented attack on any level significantly faster than exhaustive key search.
- **Message Verification:** SHA-256 of the recovered message must equal the published message hash.
- **Key Verification:** SHA-256 of "YUGA-KEY-COMMIT|" || K must equal the published key commitment; a key can also be checked against the known pairs.
- **Integrity and Closure:** The commitments fix every key and message at publication. All keys and messages will be released

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

when the challenge closes so that anyone can check them.

- **Submission:** Solutions may be sent to the author at saktheee2009@gmail.com.

VIII. LIMITATIONS

No formal bounds on differential or linear trail probabilities are given; the security claim rests on the conservative number of cycles and on public analysis. The subkeys are restricted to about 43.6 bits of range each, so the top 20 bits of every subkey are zero. However, the master key entropy is unaffected; this reduces the per-cycle key mixing in the upper bits and deserves specific analysis. The reference implementation is written for clarity, not speed, and is not constant-time; side-channel resistance is out of scope. The yuga constants are a design choice and are not claimed to add security on their own.

IX. CONCLUSION

Yuga Cipher v2 combines an S-box layer and ARX mixing under a 256-bit key, with subkey ranges and round constants drawn from the yuga time scale. Empirical tests show full diffusion after two of its 256 cycles. The five-level challenge with published commitments invites the community to measure its real security margin; we will report any break at any level in future work.

DECLARATION STATEMENT

Some of the cited references are older and are noted explicitly as [5] and [6]. However, these works remain significant for the current study, as they are pioneering in their fields.

As the article's author, I verify the accuracy of the following information.

- **Conflicts of Interest:** Based on my understanding, this article has no conflicts of interest.
- **Financial Assistance:** This article has not been sponsored or funded by any organisation or agency.
- **Ethical Approval and Consent to Participate:** The data provided in this article are exempt from the requirement for ethical approval or participant consent.
- **Data Access Statement and Material Availability:** The data in this article are publicly accessible.
- **Authors' Contributions:** The authorship of this article is contributed solely.

REFERENCES

1. J. Daemen and V. Rijmen, The Design of Rijndael: The Advanced Encryption Standard (AES), 2nd ed. Berlin, Heidelberg: Springer, 2020. DOI: 10.1007/978-3-662-60769-5
2. F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann, "Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment," in Advances in Cryptology – CRYPTO 2020, LNCS 12171, Springer, 2020, pp. 62–91. DOI: 10.1007/978-3-030-56880-1_3
3. Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," RFC 8439, IRTF, June 2018. DOI: 10.17487/RFC8439
4. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks and L. Wingers, "Notes on the design and analysis of SIMON and SPECK," Cryptology ePrint Archive, Paper 2017/560, 2017. https://eprint.iacr.org/2017/560
5. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, vol. 4, no. 1, pp. 3–72, 1991. DOI: 10.1007/BF00630563
6. M. Matsui, "Linear cryptanalysis method for DES cipher," in Advances in Cryptology – EUROCRYPT '93, LNCS 765, Springer, 1994, pp. 386–397. DOI: 10.1007/3-540-48285-7_33
7. A. Bar-On, E. Biham, O. Dunkelman and N. Keller, "Efficient slide attacks," Journal of Cryptology, vol. 31, no. 3, pp. 641–670, 2018. DOI: 10.1007/s00145-017-9266-8
8. ISO/IEC 10118-3:2018, IT Security techniques — Hash-functions — Part 3: Dedicated hash-functions (Annex C: SHA-3 extendable-output functions). Geneva: ISO, 2018. https://www.iso.org/standard/67116.html
9. M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques," NIST Special Publication 800-38A, December 2001. DOI: 10.6028/NIST.SP.800-38A

AUTHOR'S PROFILE



S. Ravichandran is an independent researcher and inventor based in Puttur, Andhra Pradesh, India (ORCID 0009-0003-6094-2534). He holds B. Com, M. Com, MA (English), MA (Economics), and M.Phil. (Economics) and LLB degrees, a postgraduate diploma in computer applications, and a Diploma in Cyber Security from Alagappa University. He served for 25 years in the Tamil Nadu Police, where he was a state gold medallist in Morse signal communication, and later taught Economics and English at the college level. Since 2014, he has pursued independent research in symmetric cryptography, focusing on block-cipher design and open cryptanalysis challenges. He is a Life Member of the Cryptology Research Society of India (CRSI) and has three published patent applications and a granted Indian design registration in AI-based cognitive and educational systems.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Publisher /Journal and/or the editor(s). The Publisher /Journal and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

APPENDIX A. CHALLENGE DATA

All values are hexadecimal; plaintexts are ASCII. Each known pair uses the same key as its level's challenge ciphertext, with the nonce shown.

A.1 Level 1 — 4 cycles (24 subkeys)

Field	Value
Challenge nonce	b371f0ebf044ca8853b127d135bfc2c1
Challenge ciphertext	7b5b83441bcb2a6ed1d919131b5ff9b25b55f1ca59da656b49c2153f2072130200b4

Field	Value
Message length	34 bytes
Message SHA-256	d2435894da01598e0d8de151599cc48b3474d9617b51ae3177a5772b8aa953a1
Key commitment	bd86cbde8478d8f147502393bfbcf12a762b2696d3228c64f92ac2a2ac073b60

Known Pair 1

Nonce	0b8600f476137f58fd70869036432f61
Plaintext	Known plaintext number one for the Yuga Cipher open challenge.
Ciphertext	c181865ebf4f181138f986e78c4c1cac07ba5f28c281961325720caface8d4c54d1d21bfde8120687360eac4dc6594d56a1049057de281599c6995345520

Known Pair 2

Nonce	afe61bf33bed1e00115cc5d1175d7d81
Plaintext	Known plaintext number two: Krita, Treta, Dvapara and Kali.
Ciphertext	ce6d62a29733ff76aa50dd374d5518716a2537a96dd9728a119303e40a7ca276447fb38aefb93a67f0b87c0f11794fcbe7818f824368e9020551d1

Known Pair 3

Nonce	39044f2673bb8f3ea1b75f6b76b616fd
Plaintext	Known plaintext three -- 1536 keys over 256 cycles.
Ciphertext	09557e44738505b97d9e8ddbe5b5d6bff2ae73f97be5b96028912fe2793d03c7befebd8f31846b8ac237fb02a42e12781a4327

Known Pair 4

Nonce	3c62ab60ad7574cce498c2926d180831
Plaintext	Known plaintext four. Break the key, win the challenge.
Ciphertext	bb213a092db1b1d74db667953316f4559e592dc5417ccae5ead25f9523206722670238fb8da6a6a4cc918a69afbc8dc1df6d5e4b018d38

A.2 Level 2 — 8 Cycles (48 Subkeys)

Field	Value
Challenge nonce	baa046481968927319cb9bb713b11116
Challenge ciphertext	754e1117f3b4ba097f3919642bfaef3885e2c3832a14aa381451e4a2bddeb8cc7a74
Message length	34 bytes
Message SHA-256	195c2bb4aa8cb373c9511070b64738770b6ff151090f41116e12206b04091e51
Key commitment	3263612a2c000113995186faa2cad048a7673da961216f512c36a59fc8c1d7a2

Known Pair 1

Nonce	85013296ee029d4873bc30940b700326
Plaintext	Known plaintext number one for the Yuga Cipher open challenge.
Ciphertext	a87dfde5fa08fc3e6530c5d682d9c4e91a784137b4a507a049bad9dcbbd991eabfe695d090469e397e3defb5b098c8d25001d01a3c9edcfb9e9f0cd42f4b

Known Pair 2

Nonce	45fc6fcab8c1d36bac1118bfb3d73c15
Plaintext	Known plaintext number two: Krita, Treta, Dvapara and Kali.
Ciphertext	5a45612cd284ee35b5ea53f775baaecb313c1697fba2740e3f4f91b136c1ed3e4fb25fe9ede703362766e34948dcf2095da185e2826da2e2bebbca

Known Pair 3

Nonce	bee9d19cc7ed7f5f2b54f77402d39f63
Plaintext	Known plaintext three -- 1536 keys over 256 cycles.
Ciphertext	7b54bcb5bcc48ece4f180bc17f5aa28207cda2e564f047c249a91732cd879057d97ebadb80a4a46399aaf425a5ca8e8acb3d3a

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Known Pair 4

Nonce	75b6881617f7d1ec90ee861e02fbb700
Plaintext	Known plaintext four. Break the key, win the challenge.
Ciphertext	ae3b4e384ce6bee0ae146e737cbcf7ba27fd415d77c59a3afca2bbf38fe76d75f86050d6f87c6ff41611fb480a5060f9f13ef65dd4da70

A.3 Level 3 — 16 Cycles (96 Subkeys)

Field	Value
Challenge nonce	662008983a143bfb49ef644890dc8b66
Challenge ciphertext	3a792e1b8fc44a16aa8171dccebfe64ea1da72417b2b42b3c5a7b9c5b39245658c034e
Message length	35 bytes
Message SHA-256	2f2cf15e4ca71d347a92cb967ac0b3cc0073f2a368d57df83b535775a7c0eb8d
Key commitment	395053c7b86868712a2d52c43e7c5c812316729a72d8ac779cea4935a1fbc1d2

Known Pair 1

Nonce	dc421fa6fb48c834a755f15170ce3bed
Plaintext	Known plaintext number one for the Yuga Cipher open challenge.
Ciphertext	8571260ff463a797f26afee30f161d770f834b633b19c320c8ab24766e1ea9d4601105d953d55c922bc7c56e43adb3d4e6ceab468f7ee029db2c97d44911

Known Pair 2

Nonce	ee6b963bd4311af10a842759b5f71f5b
Plaintext	Known plaintext number two: Krita, Treta, Dvapara and Kali.
Ciphertext	0688e17cad268540ea28c53c04ca73bbd6c94a532b494483a9deba3f5f7fb640f36aee25106d1dae809f6cceb10d5747dc7d5f3ff125a40817f084

Known Pair 3

Nonce	2121813139fdbd96a2bdcc1aced90e47
Plaintext	Known plaintext three -- 1536 keys over 256 cycles.
Ciphertext	a44acd2bba32f88f54664520cd8136780893459cb3d9c34e5910f40baddb85aa9d9d2028d38b6d6aeee52c8c9b40caa6bc0dcf

Known Pair 4

Nonce	5375d18cfaf6a06462245fa6a83b74c1
Plaintext	Known plaintext four. Break the key, win the challenge.
Ciphertext	e4f3b09e64e6e9a1dbdf055d3247b3f1e2532209e7b030f7a09e8686b0024a4da63588408e5635fe6903c4e0618b91676e5e81152ce234

A.4 Level 4 — 64 Cycles (384 Subkeys)

Field	Value
Challenge nonce	57610dac041a5d0da3fd064ad179f2a1
Challenge ciphertext	89664230fe470721597e7c5e2a9683a223d3de937ad2d0fe7bd17d5cea821f32e1ab25
Message length	35 bytes
Message SHA-256	f83ab48f49d0aca555b848df8d5d9737bc9269eaf68fe484132fece8b28af920
Key commitment	da78a6faaad9938b37274f40b8ecc810c87fcb0537e144ffb768dfad154b866e

Known Pair 1

Nonce	17b970568081221e168483c46b3fd666
Plaintext	Known plaintext number one for the Yuga Cipher open challenge.
Ciphertext	927c4bcda281bbc9eae53bd0258d376449875f844a2b0c8915c49433d47b6c399a36fd08290fe603503f73f2d9b9cd7da151861d042fa7756243304b28f0

Known Pair 2

Nonce	081e2eae0f63de16d20bc085c65bdb8f
Plaintext	Known plaintext number two: Krita, Treta, Dvapara and Kali.
Ciphertext	d2fbf058826e9446967afe0566b020a758c2e886ed875ce90ceac60b8c8ba4a032a33a2ff676357b6ecda0cddc49b88a22c4578f0169a23aed3d63

© Copyright: All rights reserved.

Known Pair 3

Nonce	20aa10d1c4de1b4b18b226fff5139ca7
Plaintext	Known plaintext three -- 1536 keys over 256 cycles.
Ciphertext	6899c931ce7e47b024d5c454eefb35de2ea19a737352ddac475ec4b3b25aca8d940569dbe26dbd68a0e1f9c07aa274522b22c8

Known Pair 4

Nonce	ecc6e5f1e077e38160af71a6d0802b5f
Plaintext	Known plaintext four. Break the key, win the challenge.
Ciphertext	1061824101307d1fd32cdccce07bd7251d72854aa20267f5f1c54ad8b5790a44093c0902d938a0793507c67eeae2bc3597f42129ee0b37

A.5 Level 5 — 256 Cycles (1536 Subkeys)

Field	Value
Challenge nonce	fd81e55328d700b9f4debb1d4d20e87e
Challenge ciphertext	40eb644e1e63ea6dafe054c4d6df9d309d7575a75a341aba8c61d8bb27060273758c55f535e783ee9d444da8d7a83324c2b5
Message length	50 bytes
Message SHA-256	df4bfedd394d8505747f8698d82469a0afc415868aae3da86fc53bb4316e702f
Key commitment	64e5cc48a80c2fba8cfbc294c30d4101d8f11ee55e3c97754b9a1d7c9314b274

Known Pair 1

Nonce	82c6932a07e77918169a319b7bb25ee6
Plaintext	Known plaintext number one for the Yuga Cipher open challenge.
Ciphertext	f83c1bb636b8fd13d10bc59a11022f1b5a83d74a34d7692acc4959a5a98489e248a8c269f521e13f4eb5a1e06b7faec09b899d3e16a56fe5cc64dd9de501

Known Pair 2

Nonce	c0a8bd11b5600bc62d9f52801aa9f5d0
Plaintext	Known plaintext number two: Krita, Treta, Dvapara and Kali.
Ciphertext	c888eca01942a030dad023b7a2067b348e0ea474172fc45eb40c1e27b648b516970960dbab62e1a9e3429172ee3b268de94da86baf0957f01923f9

Known Pair 3

Nonce	0bf94089ea2b97dc7da1755110e54ea5
Plaintext	Known plaintext three -- 1536 keys over 256 cycles.
Ciphertext	266c968059164488fcdd483753e356d26d3a4fe8d93afe8e1dbefa7200916aff5bb0e13dc0991d44e69a17a6611a60c476c998

Known Pair 4

Nonce	d3cc5260c57cd103fa6f7f80faafab94
Plaintext	Known plaintext four. Break the key, win the challenge.
Ciphertext	145c7dbe812f709680209183073096cae34789bd3f690efa67e1b936fb1e1d7389bddee1dc80d26a0deb75b5ee860a62d36f7e5c1a8fb9

APPENDIX B. REFERENCE IMPLEMENTATION

Python 3, standard library only.

```
"""
YUGA CIPHER v2-- PUBLIC ALGORITHM (Open Challenge 2)
Author: S. Ravichandran

Everything in this file is public. Only the 256-bit master key is secret
(Kerckhoffs's principle).

Design summary
--------------
* Master key: 256 bits (32 bytes), chosen uniformly at random.
* Key schedule: master key -> 1536 subkeys (256 cycles x 6 keys) using
                SHAKE-256. Every subkey is a whole number of SECONDS
                inside ONE YUGA (0 <= k < 4,20,000 years in seconds),
                selected by rejection sampling so all values are equally
                likely.
* Round constants: taken from the FOUR-YUGA span (17,20,000 years in
                seconds): C_r = (r+1) * FOUR_YUGA_SECONDS, mixed to 64 bits.
* Block: 128 bits held as two 64-bit words A, B.
* One cycle: 6 subkeys, byte-wise 8-bit S-box (non-linear layer),
                add-rotate-xor (ARX) mixing, round constant, word swap.
```


© Copyright: All rights reserved.

```
* Cycles: 256 (full). Reduced-cycle versions use the first N cycles.
* Mode: CTR (counter) mode with a public 128-bit nonce, so any
        message length works:  C = P XOR E_K(nonce + counter).

Security claim (to be tested by the community):
   The best attack should need about 2^256 work (key search).
   Real key space = 2^256 because the master key is 256 random bits;
   the 1536 subkeys are derived from it, so they add no extra key space.
"""

import hashlib

YEAR_SECONDS = 365.25 * 24 * 60 * 60                # Julian year
ONE_YUGA_SECONDS = int (420_000 * YEAR_SECONDS) # 13,254,192,000,000
FOUR_YUGA_SECONDS = int (1_720_000 * YEAR_SECONDS) # 54,279,072,000,000

FULL_CYCLES = 256
KEYS_PER_CYCLE = 6
MASK64 = (1 << 64) - 1

# 8-bit S-box (the AES S-box: public, well-studied non-linear layer)
SBOX = [
   0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76,
   0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0,
   0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15,
   0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75,
   0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84,
   0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf,
   0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8,
   0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2,
   0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73,
   0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb,
   0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79,
   0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08,
   0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a,
   0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e,
   0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf,
   0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16,
]


def rotl(x, n):
   return ((x << n) | (x >> (64 - n))) & MASK64

def sub_bytes(x):
   """ Apply the 8-bit S-box to each of the 8 bytes of a 64-bit word."""
   out = 0
   for i in range (8):
      out |= SBOX [(x >> (8 * i)) & 0xFF] << (8 * i)
   return out

 def round_constant(r):
   c = ((r + 1) * FOUR_YUGA_SECONDS) & MASK64
   c ^= rotl (c, 23) ^ rotl (c, 47)
   return c

 def key_schedule(master_key: bytes, cycles: int = FULL_CYCLES):
   """Master key (32 bytes) -> cycles x 6 subkeys, each in [0, ONE_YUGA_SECONDS)."""
   if len(master_key) != 32:
      raise ValueError("master key must be exactly 32 bytes (256 bits)")
   need = cycles * KEYS_PER_CYCLE
   limit = (1 << 44)                    # ONE_YUGA_SECONDS < 2^44
   xof = hashlib. shake_256(b"YUGA-CIPHER-v2|" + master_key)
   subkeys, buf_len = [], need * 8
   while len(subkeys) < need:
      stream = xof.digest(buf_len)
      subkeys = []
      for i in range (0, len(stream), 8):
         v = int.from_bytes(stream [i:i + 8], "big") & (limit - 1)
         if v < ONE_YUGA_SECONDS:       # rejection sampling -> uniform
            subkeys.append(v)
            if len(subkeys) == need:
               break
      buf_len *= 2
   return [subkeys [i:i + 6] for i in range (0, need, 6)]

 def encrypt_block(block16: bytes, schedule) -> bytes:
   A = int.from_bytes(block16[:8], "big")
   B = int.from_bytes(block16[8:], "big")
   for r, (k0, k1, k2, k3, k4, k5) in enumerate(schedule):
      A = (A + k0) & MASK64
      B ^= k1
      A, B = sub_bytes(A), sub_bytes(B)             # non-linear layer
      A = (A + rotl (B, 13)) & MASK64               # ARX mixing
      B = rotl (B ^ A, 29)
      A ^= k2
      B = (B + k3) & MASK64
```


© Copyright: All rights reserved.

```
        A = sub_bytes(A)
        B ^= rotl (A, 41)
        A = (A + B) & MASK64
        A ^= k4 ^ round_constant(r)
        B ^= k5
        A, B = B, A                              # swap halves
    return A.to_bytes(8, "big") + B.to_bytes(8, "big")

 def ctr_crypt(data: bytes, master_key: bytes, nonce: bytes,
              cycles: int = FULL_CYCLES) -> bytes:
    """ Encrypt or decrypt (same operation) in CTR mode."""
    if len(nonce) ! = 16:
        raise ValueError("nonce must be 16 bytes")
    schedule = key_schedule(master_key, cycles)
    n = int.from_bytes(nonce, "big")
    out = bytearray()
    for i in range (0, len(data), 16):
        ctr = ((n + i // 16) % (1 << 128)).to_bytes(16, "big")
        ks = encrypt_block(ctr, schedule)
        chunk = data [i:i + 16]
        out += bytes (a ^ b for a, b in zip (chunk, ks))
    return bytes(out)

# ---------- public test vector (key is NOT secret here) ----------
TEST_KEY = bytes (range (32))
TEST_NONCE = bytes (16)
TEST_PLAINTEXT = b"Yuga Cipher v2 test vector."

if __name__ == "__main__":
    ct = ctr_crypt(TEST_PLAINTEXT, TEST_KEY, TEST_NONCE)
    print ("Test key       :", TEST_KEY.hex())
    print ("Test nonce     :", TEST_NONCE.hex())
    print ("Test plaintext:", TEST_PLAINTEXT.decode())
    print ("Test ciphertext:", ct.hex())
    assert ctr_crypt(ct, TEST_KEY, TEST_NONCE) == TEST_PLAINTEXT
    print ("Decryption check: OK")
```


© Copyright: All rights reserved.