🔺↕ 512×512 Inverted Hash

> A novel cryptographic hash function based on a **pyramidal reduction architecture**
> with **Sacred Pair inversion** at dimension 257 — the K1 ↔ K2 boundary.
>
> **Author:** Marco Lazcano — Decatur, GA, USA — roble99oak77@gmail.com — May 2026

⚠️ Security Notice

This is a **research prototype**. It has not been formally audited.
Do not use in production systems or cryptographic protocols
without independent peer review and formal cryptanalysis.

Overview

Property	Value
Input block	256 bytes
Internal state	256 bytes (2048 bits)
Output	256 bytes (2048 bits)
Permutation rounds	24
Chunk size	256 bytes (full state, one block)
S-box	AES S-box (NL=112)
Round constants	16 × 32-bit values from π
Inversion	Sacred Pair SP(x) = (~x & 0xFF)
K1/K2 boundary	Dimension 257
Architecture	Sponge construction

Statistical Test Results — Full 2048-bit Output

All 11 tests on the **complete 256-byte (2048-bit) output**.
N=1,000 samples for Pearson correlation — statistically definitive.

Test	Result	Threshold	Status
Basic functionality	All distinct	All distinct	✅
Determinism (200 trials)	0 errors	0 errors	✅
Avalanche mean	**49.906%**	50.00% ± 2%	✅ EXCELLENT

Std deviation	**1.104%**	< 5%	✅ EXCELLENT
Shannon entropy	**7.9974 b/byte**	> 7.98	✅ EXCELLENT
Entropy efficiency	**99.97%**	> 99%	✅ EXCELLENT
Chi-square	**237.51**	< 293.2 (95%)	✅ EXCELLENT
Pearson corr. avg (N=1000)	**0.02524**	< 0.05	✅ EXCELLENT
Pairs with corr > 0.10	**52 / 32,640** (0.16%)	~0%	✅ EXCELLENT
S-box max differential	**1.56%**	< 6.25%	✅ SECURE
SAC deviation > 10%	**1.09%**	< 5%	✅ EXCELLENT
Collision resistance	**0 / 400**	0	✅ ROBUST
Sacred pair properties	Verified	Involution + XOR	✅

Design Evolution: v1 → v2

The initial implementation revealed a diffusion limitation.
Two architectural fixes resolved it completely:

Metric	v1 (chunk=64, 12 rounds)	v2 (chunk=256, 24 rounds)
Avalanche mean	30.94% ⚠️	**49.906%** ✅
Std deviation	13.63% ⚠️	**1.104%** ✅
SAC deviation >10%	75.27% ⚠️	**1.09%** ✅

Fix A: Full 256-byte state processed as one pyramidal block
(was: four 64-byte chunks — insufficient cross-chunk diffusion).

Fix B: 24 rounds (was: 12 — insufficient for complete mixing).

How It Works

1. Pyramidal Nonlinear Function

The full 256-byte state is iteratively compressed level by level:

```python
diff = SBOX[(seq[i+1] - seq[i]) mod 256]   # AES S-box substitution
diff = safe_byte(diff XOR anchor)           # anchor mixing
diff = rotate_left(diff, (i mod 7) + 1)     # position-dependent rotation
```

This continues until a single byte remains — the pyramid tip.

2. Sacred Pair Inversion — K1 ↔ K2

```python
SP(x) = (~x) & 0xFF    # bitwise complement

# Applied at the K1/K2 boundary (dimension 257):
inverted[i] = SP(anchor)  if i >= 256  else anchor
```

Properties of Sacred Pair:
- Involution: `SP(SP(x)) = x` for all x ∈ [0, 255]
- Maximum XOR distance: `x XOR SP(x) = 0xFF` for all x

With standard 256-byte inputs, the pyramid produces 255 anchors
(i = 0..254), so the K2 region is a **structural boundary** —
a reserved extension for future bidirectional architectures.

3. Internal Permutation (24 rounds)

Each round applies three steps to the full 256-byte state:

1. **Full-state pyramidal mixing** — entire 256-byte state as one block
2. **Linear diffusion** — each byte XOR-ed with the next
3. **Nonlinear substitution** — AES S-box on every byte

4. Padding & Absorption

- Append `0x80`
- Zero-pad until `len(buffer) mod 256 == 248`
- Append message length in bits as 64-bit little-endian integer

---

Installation

No external dependencies. Pure Python 3.6+.

```bash
git clone https://github.com/your-username/inverted-hash-512x512.git
cd inverted-hash-512x512
python3 inverted_hash_512x512_analysis.py
```

Usage

```python
from inverted_hash_512x512 import InvertedHash

# Full 2048-bit output (default)
h = InvertedHash()
h.update(b"Hello, world!")
print(h.hexdigest(256))    # 512 hex chars = 2048 bits

# 256-bit output
h2 = InvertedHash()
h2.update(b"Hello, world!")
print(h2.hexdigest(32))    # 64 hex chars = 256 bits

# Streaming
h3 = InvertedHash()
h3.update(b"first chunk")
h3.update(b"second chunk")
print(h3.hexdigest(256))
```

Example outputs (first 32 hex chars)

```
b"" (empty)       ->  36ede6bdc0eeff49375d94935936cf52...
b"\x00"           ->  ecccf4678a7b4075e67a8fd4f9433954...
b"\x01"           ->  95feb8f924c64adfe1b6bc724cd741d3...
b"\xff"           ->  915679b2a4a0cfbccdcfef01eef5a297...
b"Hola Mundo"     ->  7810e0e497bd6e29a186fb12129175d9...
b"Hello"          ->  84d74494575458ff0677f0119bc26b70...
b"hello"          ->  00b9bff83f086e9b8d1d5e7d0c7a168d...
b"hello " (space) ->  f5dfb7b95923739a1aa7a11472de140b...
```

Design Decisions

Component	Choice	Rationale
S-box	AES S-box (unchanged)	NL=112, max differential 1.56%

Round constants	Digits of π	Nothing-up-my-sleeve
Architecture	Sponge construction	Formal security guarantees
State size	256 bytes	1024-bit birthday bound
Chunk size	256 bytes (full state)	Complete cross-state diffusion
Rounds	24	Complete diffusion across 2048 bits
Inversion	SP(x) = ~x & 0xFF	Involution with max XOR distance

Performance

> Pure Python 3 — not representative of a compiled implementation.

Input size	Time / hash
256 bytes	~1,444 ms
512 bytes	~1,924 ms
1 KB	~2,885 ms
4 KB	~8,658 ms

A C/Rust implementation is estimated to be ~100× faster.

Reproducing the Tests

```bash
# Full analysis — all 11 tests
python3 inverted_hash_512x512_analysis.py

# Note: Pearson correlation (N=1000) takes ~25 min
# Note: SAC (150 samples, 8 bits) takes ~14 min
# Total runtime: ~45-50 minutes
```

Comparison with Pyramidal Hash 256×512

Both designs by Marco Lazcano (May 2026):

Metric	Pyramidal Hash 256×512	**Inverted Hash 512×512**
Output bits	512	**2048**

Avalanche	50.54% σ=2.26%	**49.906% σ=1.104%**
Entropy	7.9914 b/byte	**7.9974 b/byte**
Chi-square	275.30	**237.51**
Corr. avg	0.1848 (N=150)	**0.02524 (N=1000)**
SAC	not measured	**1.09%**
Novel feature	Pyramidal reduction	**Sacred Pair K1↔K2**

Known Limitations

- **No formal cryptanalysis** — differential, linear, algebraic attacks not studied.
- **K2 never activated** — with 256-byte inputs, i≥256 is a structural boundary.
- **Python only** — unsuitable for production performance.
- **No external audit** — cryptographic peer review pending.

Academic Paper

📄 **512×512 Inverted Hash: A Novel Cryptographic Hash Function Based on Pyramidal Reduction with Sacred Pair Inversion**
Marco Lazcano, May 2026
(submitted for peer review — link will be added upon publication)

See also: **Pyramidal Hash 256×512** — the companion design by the same author.

Contributing

Contributions especially welcome for:

- 🔬 Formal cryptanalysis (differential, linear, algebraic)
- 🔑 K2 activation — exploring inputs with > 256 pyramid anchors
- ⚡ C / Rust implementation
- 🧪 NIST SP 800-22 test suite

License

References

1. Daemen & Rijmen (2002). *The Design of Rijndael: AES*. Springer.
2. Bertoni et al. (2011). *Cryptographic sponge functions*. IACR.
3. NIST (2015). *SHA-3 Standard*. FIPS PUB 202.
4. NIST (2012). *SHA-2 Standard*. FIPS PUB 180-4.
5. Menezes et al. (1996). *Handbook of Applied Cryptography*. CRC Press.
6. Lazcano, M. (2026). *Pyramidal Hash 256×512*. Independent Research.
