A Cryptographic Note on the Rainstorm Hash Function

(Cris)

December 17, 2024

Introduction

Rainstorm is a proposed hash function featuring a 1024-bit internal state S that processes data in 512-bit blocks. Its design deliberately blends concepts from multiple well-established cryptographic paradigms. Rather than adhering strictly to Merkle-Damgård, Feistel, sponge, or ARX constructions, Rainstorm borrows from each of these ideas, forging a hybrid approach. Instead of the standard ARX (Add/Rotate/XOR) pattern, Rainstorm uses XSR (XOR, Subtract, Rotate), where subtraction mod 2^{64} replaces addition. This subtle shift maintains fast, hardware-friendly mixing while introducing a slightly different algebraic structure.

Drawing inspiration from iconic designs like DES and Blowfish, Rainstorm also employs carefully chosen constants and rotation amounts, echoing the classical approach of using round-dependent parameters to maximize diffusion. By merging multiple paradigms and tuning parameters with the help of statistical testing tools like SMHasher3, Rainstorm seeks to explore new design space in hash functions—potentially offering unique security properties if thoroughly validated.

However, Rainstorm's complexity and unconventional blend of methods mean its true security remains unproven. This note outlines the motivation behind this hybrid design, examines its relationship to established primitives, details its mixing strategy, and discusses areas where cryptanalysis is needed. It encourages cryptographers to investigate whether Rainstorm can withstand modern attacks and what lessons can be drawn from its construction.

Motivation: Why Merge These Paradigms?

Over decades of cryptographic research, different approaches to hashing and encryption have emerged, each with its own strengths and weaknesses:

- **Merkle-Damgård (MD) Hashes:** While MD5, SHA-1, and similar schemes are now broken or weakened, the iterative compression approach they pioneered remains conceptually elegant. Processing the message in fixed-size blocks, updating an internal state iteratively, and then producing a final digest is a well-understood paradigm that ensures every part of the input message influences the final hash.
- **Feistel Networks (e.g., DES, Blowfish):** Feistel constructions achieve balanced diffusion by splitting the state into two halves and alternating which half is transformed each round. DES and Blowfish demonstrated how clever choice of round functions and constants can complicate cryptanalysis, and showed that even a relatively "weak" round function, when iterated enough times, can yield a strong overall design. Borrowing this half-state alternation can enhance desirable diffusion properties even in a hash setting.

- **Sponge Constructions (e.g., Keccak/SHA-3):** The sponge framework's absorb-then-permute-then-squeeze model provides flexibility, extendable outputs, and robust security properties when well-designed. While Rainstorm is not a full sponge (it doesn't follow the same padding rules or full permutation structure), it can mimic certain sponge-like behaviors, such as absorbing input and optionally re-injecting output as new input for arbitrary-length outputs.
- **XSR (XOR, Subtract, Rotate) Operations:** Classic ARX primitives (e.g., ChaCha, Salsa20) rely on repeated application of a "weak" round function to achieve complexity. Rainstorm's XSR approach similarly depends on iteration. By using subtraction instead of addition, it introduces a twist that still disrupts linear patterns efficiently, while retaining ARX's strength of simplicity and speed. This provides nonlinearity and efficient mixing without the complexity of S-boxes. Combined with chosen rotation amounts and subtraction constants, Rainstorm's XSR aims to spread input differences widely across the state.

By blending these ideas, Rainstorm attempts a defense-in-depth strategy. Breaking it might require overcoming multiple structural hurdles simultaneously. While no single paradigm guarantees security—MD constructs were broken, DES is outdated, and ARX-only ciphers have known forms of analysis—the hope is that combining their "best" aspects and carefully chosen constants yields a more resistant design.

The Use of Iterated Weak Functions

Many well-known designs rely on repeating a relatively simple or *"weak"* round function multiple times to achieve cryptographic strength. For example:

- **Block Ciphers Like DES and Blowfish:** Each round's transformation (the Feistel function) is not individually cryptographically strong. However, applying it repeatedly (16 rounds in DES, variable in Blowfish) compounds complexity and makes the final construction secure enough for its era and key sizes.
- **Sponge-Based Functions (Keccak/SHA-3):** The Keccak permutation is a repeated application of a simple, *"weak"* round. A single round of Keccak is not considered secure by itself, but multiple rounds produce a strong permutation.
- **ARX-Based Designs (e.g., ChaCha):** ChaCha's quarter-round function is simple, but applying it many times in a structured way yields a strong stream cipher.

Rainstorm follows this common pattern: it defines a simple mixing step (weakfunc) that is not individually secure, but repeated sufficiently many times—guided by empirical testing—results in a well-mixed, statistically strong output. By combining elements of MD iteration, Feistel mixing, sponge-like absorption, and XSR operations, Rainstorm aims to create complexity through iteration, much like these other established designs.

Guided by Statistical Testing (SMHasher3)

Rainstorm's development was iterative and guided by SMHasher3, a suite of statistical tests for hash functions. Through trial and adjustment, the designer found that at least four rounds of its mixing function are needed to pass all 238 SMHasher3 tests. One round fails many tests, two or three rounds still show some weaknesses, but at four rounds every test passes, indicating strong statistical mixing properties.

This iterative tuning reveals a knob for balancing performance and security:

- **Fewer Rounds:** Faster but weaker statistically.
- **More Rounds:** Slower but more complex and potentially more secure.

While passing SMHasher3 does not guarantee cryptographic strength, it suggests that Rainstorm's chosen constants, rotation amounts, and iteration count produce a well-mixed output distribution. This empirical approach can inspire cryptanalysts to investigate whether these statistical properties hold up under adversarial attacks.

Drawing Inspiration from Established Constructions

Rainstorm's structure is influenced by multiple paradigms:

Merkle-Damgård-like Iteration

Like MD-based schemes (e.g., SHA-2), Rainstorm processes the input message in blocks. Each block updates the internal state S, ensuring all parts of the message contribute to the final hash. Even though pure MD-based hashes like MD5 are compromised, the iterative idea remains a solid foundation for incrementally absorbing large messages.

Feistel-like Internal Mixing

Rainstorm splits its 1024-bit state into two 512-bit halves, updating one half at a time while referencing the other. This is reminiscent of Feistel networks like DES and Blowfish, where each round transforms one half and then swaps. Rainstorm's approach ensures that differences introduced in one half propagate thoroughly into the other over successive rounds. By carefully choosing constants and rotation schedules—somewhat like the fixed S-boxes and key schedules in DES or Blowfish—Rainstorm aims to achieve robust diffusion.

Sponge-Inspired, Not Full Sponge

Although not a full sponge (it lacks formal sponge padding and rate/capacity division), Rainstorm's large state and iterative round function allow it to absorb message data and potentially continue "absorbing" its own output to produce longer results, a behavior analogous to an extendable-output function (XOF). This sponge-inspired flexibility means that Rainstorm could be adapted to produce variable-length outputs, opening doors to uses beyond standard fixed-size hashing.

XSR for Mixing Without S-boxes

Replacing addition with subtraction doesn't reduce efficiency. XOR and rotation, staples of ARX designs, remain. This slight twist—XSR—maintains speed and simplicity, introducing just enough variation to potentially resist ARX-focused attacks. Without fixed S-boxes, Rainstorm relies on carefully chosen constants and rotation amounts to ensure avalanche and disrupt patterns, similar to how Blowfish used carefully-crafted S-boxes and rotation strategies.

Tuning and Development Methodology

Rainstorm's parameters, such as chosen constants and the number of rounds, were guided by iterative testing with tools like SMHasher3. SMHasher3 subjects the hash output to a wide range of statistical tests. Empirically, Rainstorm found that four rounds were sufficient to pass all tests with the chosen constants, while fewer rounds failed some tests.

This suggests a tunable security parameter. One round isn't secure, two or three struggle with some tests, but at four rounds we get robust statistical properties. Increasing rounds presumably increases security margins, at the cost of performance.

From Classical Primitives to Rainstorm: Side-by-Side Comparisons

Comparing Rainstorm with well-known primitives:

Merkle-Damgård vs. Rainstorm

Merkle-Damgård Step:

$$S \leftarrow f(S, M_i)$$

Repeat per block.

Rainstorm Step:

$$S \leftarrow F(S, M_i, \text{constants})$$

Repeat per block.

While Merkle-Damgård uses a traditional compression function f, Rainstorm's F uses XSR transformations to incorporate message blocks into a large state, blending old concepts with fresh ideas.

Feistel-Like Alternation in Rainstorm

Feistel Step:

$$S = (L||R)$$

$$L \leftarrow R$$

$$R \leftarrow L \oplus F(R)$$

Rainstorm Version:

$$S = (S_L||S_R)$$

$$S_L \leftarrow F(S_L, M, K, Z)$$

$$S_R \leftarrow S_R \oplus S_L$$

Alternating which half is updated each round helps spread differences throughout the state, much like a Feistel network.

XSR Operations in Rainstorm

Classical ARX:

$$S \leftarrow (S + M) \oplus R$$

Rainstorm (XSR) Version:

$$S_L \leftarrow (S_L \oplus M) - K$$

$$S_L \leftarrow \text{ROTR64}(S_L, Z)$$

XSR provides a similar lightweight, hardware-friendly way to achieve non-linearity and confusion.

Detailed Examination of the Weak Function (weakfunc)

The heart of Rainstorm's design is its "weak function" applied multiple times per block. While a single application is weak, repeated iterations build complexity. The weak function integrates the message block M, a set of constants K, rotation schedules Z, and the internal state halves S_L, S_R:

Listing 1: Rainstorm Mixing Function (Conceptual)

```
def F(S, M, K, Z):
    S_L, S_R = split(S)

    # Ingest M into S_L, then subtract K, then rotate
    S_L = (S_L ^ M) - K
    S_L = ROTR64(S_L, Z)

    # Blend updated S_L into S_R
    S_R = S_R ^ S_L

    return combine(S_L, S_R)
```

This simplified pseudocode omits details like counters and "folding." The actual implementation uses counters and conditions depending on whether we're updating the "left" half or the "right" half, mimicking a Feistel-like pattern. Counters ensure that each round differs slightly, preventing the state from falling into predictable cycles.

Folding, Counters, and Constants

Folding certain values back into the state and employing counters after each update add complexity. These steps prevent stable differentials from easily forming. The constants K and rotation amounts Z were chosen to produce good avalanche properties, informed by trial, testing, and lessons from older ciphers.

This approach echoes the principle used in DES and Blowfish, where constants and rotation schedules were deliberately chosen (and sometimes derived from known mathematical constants or tested sets) to ensure no simple linear or differential structure emerges.

Influence of SMHasher3 on Parameter Choice

Parameters like the number of rounds and the selection of constants weren't chosen arbitrarily. Through repeated testing with SMHasher3's extensive randomness tests, Rainstorm's designer found that fewer than four rounds produce statistically detectable patterns. At four rounds, all tests are passed, suggesting that the combination of XSR operations, chosen constants, and Feistel-like alternation has reached a threshold of complexity.

While SMHasher3 doesn't test for deep cryptanalytic strengths (like resistance to differential cryptanalysis or linear attacks), it does ensure there are no obvious statistical biases. This empirical feedback loop—test, tweak constants, retest—guided the refinement of Rainstorm's parameters.

Cryptanalysis Considerations and Open Questions

Despite passing all SMHasher3 tests at four rounds, Rainstorm's true cryptographic strength remains unproven. Questions include:

- **Round Count:** Four rounds suffice to pass SMHasher3, but is that enough for cryptographic security? Would an attacker find shortcuts at fewer rounds or extended attacks that apply at four rounds?
- **Differential and Rotational Attacks:** Feistel-like structures and ARX designs are known battlefields for differential cryptanalysis. XSR may introduce new complexities. Can subtle patterns survive multiple rounds? Can an attacker find a differential trail that propagates through the XSR operations and the alternating halves? Are there rotation-based patterns that survive multiple rounds, or subtle algebraic relationships that could linearize the state updates?
- **Linear and Algebraic Attacks:** Without S-boxes, linear approximations may be easier or harder—this depends on how subtraction and rotation interact algebraically. ARX primitives have known strategies for linear approximations, and while XSR might differ slightly, could similar methods apply? Are there hidden structures amenable to linearization or algebraic factorization?
- **Reduced-Round Analysis:** Analyzing a three-round or even two-round variant might reveal structural weaknesses that are masked at four rounds. Understanding how security degrades when we reduce rounds can guide recommendations for practical security margins.

Why Analyze Rainstorm?

In a landscape where SHA-3, BLAKE3, and other well-established hashes dominate, why spend time on an outsider's hybrid design?

- **Fresh Cryptanalytic Challenge:** Rainstorm's hybrid nature—MD-like iteration, Feistel alternation, sponge inspiration, and XSR mixing—presents a less familiar target. Traditional attacks tailored to pure ARX or classical Feistel might not apply directly. This encourages cryptanalysts to develop or adapt their techniques, potentially advancing the field.
- **Testing Old Assumptions:** If Rainstorm resists certain known attacks that easily break simpler schemes, it might validate the idea of mixing paradigms. Conversely, if cryptanalysts find a shortcut, we learn more about the limitations of blending these methods and where hidden vulnerabilities lie.
- **Guiding Future Designs:** Understanding Rainstorm's strengths and weaknesses can inform future hash designs. Even if Rainstorm never becomes a standard, insights from its analysis could help designers avoid its pitfalls or incorporate its successful elements.
- **Performance and Practical Trade-offs:** Rainstorm is relatively fast and can be tuned by adjusting the number of rounds. If cryptanalysis shows that four rounds already offer decent resistance, Rainstorm could find niche applications where both speed and complexity matter.

Comparative Positioning and Intended Audience

Rainstorm is experimental, probing whether blending elements from classical paradigms and leveraging an iterated weak function can yield novel resilience or highlight new weaknesses. It's a conceptual prototype—an "academic outsider" experiment that combines known paradigms in a novel way. Its intended audience includes:

- **Cryptanalysts:** Researchers who want to test their skills against a new construction that doesn't fit neatly into well-known categories.
- **Protocol and Primitive Designers:** Individuals looking for design inspiration, curious about whether mixing these paradigms can yield a secure, high-performance function.
- **The Broader Crypto Community:** Anyone interested in exploring how established ideas can be recombined, refined, or challenged to inspire the next generation of hashing strategies.

FAQ and Additional Questions

Q: Why create a new hash when SHA-3 and BLAKE3 are well-established? A: Rainstorm is experimental. It probes whether blending elements from Merkle-Damgård, Feistel, sponge-like absorption, and XSR mixing yields new forms of resilience—or reveals fundamental weaknesses. Studying Rainstorm can provide lessons that inform future designs.

Q: Is it production-ready? A: No. Without extensive cryptanalysis, it's risky. Think of Rainstorm as a concept car, not a commuter vehicle. It's meant to inspire study, not secure your data today.

Q: How do I tune its security? A: Increase the rounds. Four rounds pass all SMHasher3 tests, but more rounds could raise the security margin. Adjusting parameters and testing again can help find an optimal balance between speed and complexity.

Q: Is it a sponge? A: Not formally. It's sponge-inspired but lacks the full structure and padding rules. It can, however, absorb input and re-inject output to produce extended results, mimicking sponge-like behavior in a less formal manner.

Conclusion

Rainstorm creatively synthesizes concepts from Merkle-Damgård iteration, Feistel mixing, sponge-like absorption, and XSR-based operations. Like many known ciphers and hashes that rely on iterating a weak round function—DES, Blowfish, Keccak, ChaCha—Rainstorm bets that complexity emerges from repetition rather than individual round strength. It synthesizes ideas from multiple sources: it processes messages iteratively like MD-based hashes, mixes halves Feistel-style, takes cues from sponge architectures for flexibility, and uses XSR operations (instead of ARX) to achieve non-linear mixing without S-boxes. Its constants and rotations are chosen with an eye toward maximizing diffusion, recalling the design principles behind DES and Blowfish.

Guided by SMHasher3 tests, its parameters are tuned to show no obvious statistical weaknesses at four rounds. Yet true cryptographic security demands careful scrutiny. Will cryptanalysts find hidden linear or differential structures? Will fewer rounds be easily compromised?

If Rainstorm breaks under analysis, the community gains insights into what not to do. If it endures, it validates mixing paradigms and might inform future designs. Either outcome enriches our understanding, fosters dialogue, and encourages the exploration of hybrid strategies in cryptographic hash functions.