Mango Documentation

# 1. Introduction

**1.1 What is Mango?**

Mango is a next-generation cryptographic platform designed to explore and surpass the capabilities of traditional encryption standards like AES. It provides a modular workbench for constructing, evaluating, and optimizing encryption sequences made from atomic transform operations. By embracing adaptive logic, deep scoring metrics, and exhaustive search techniques, Mango enables the discovery of encryption sequences tailored to specific data types—unlocking performance and security improvements that static ciphers cannot achieve.

**1.2 Philosophy & Goals**

Mango is built on the belief that cryptography should be:  
  
- Transparent – Every decision, from transform logic to scoring outcomes, is measurable and explainable.  
- Modular – Small, composable units (transforms) empower rapid experimentation and evaluation.  
- Adaptive – Rather than applying a one-size-fits-all cipher, Mango discovers optimal sequences based on the input data's nature.  
- Provable – All sequences are evaluated with rigorous metrics, ensuring repeatable results across multiple test conditions.  
  
The overarching goal is to create a flexible, cryptographically sound, and verifiably superior encryption system that evolves with the data it protects.

**1.3 Key Features**

- 🔐 Atomic Transform Engine – Over 30 customizable transforms enable fine-grained control over data processing.  
- 🧠 Munge (Sequence Discovery Engine) – Automatically identifies top-performing sequences through multi-level, multi-metric optimization.  
- 📊 Metric-Driven Scoring – Evaluates sequences on up to 9 independent metrics, including Avalanche, Frequency Distribution, Key Sensitivity, and more.  
- 🧬 Input-Aware Adaptation – Supports data classification (Natural, Random, Sequence, Combined) to tailor sequences for maximum effectiveness.  
- ♻️ Fully Reversible Design – Ensures all sequences and transforms are invertible and validated for correctness.  
- 🚀 AES Benchmarking – Built-in comparative tools highlight where Mango outperforms AES in both strength and speed.

# 2. Quick Start

2.1 Running Your First Sequence  
The Mango Workbench allows you to define and execute sequences of cryptographic transforms. Here's how to run your first sequence:

1. Selecting Transforms:

* Transforms are identified by number (e.g., 1, 2, 3...) and each number corresponds to a specific transform name (e.g., ApplyMaskBasedMixingTx).
* To add a transform to the *current sequence*, type its number and press Enter.
* You can add multiple transforms to the sequence by entering their respective numbers, one after the other. You can also add the same transform multiple times.

1. Executing the Sequence:

* Once you have defined your desired sequence of transforms, enter the command run sequence.

1. Managing Sequences:

* To clear the *current sequence* and start a new one, enter the command clear sequence.
* You can also quickly load a previously defined sequence by entering a dollar sign ($) followed by the sequence. For example:  
  Enter your choice or command: $PatternEqualizerTx ShuffleNibblesFwdTx BitRandFlipTx
  + The *current sequence* will be updated to:  
    Current Sequence: PatternEqualizerTx(ID:28)(TR:1) ShuffleNibblesFwdTx(ID:18)(TR:1) BitRandFlipTx(ID:2)(TR:1)
    - (Note: The (ID:...) and (TR:...) notations indicate the transform's ID and transform rounds.)

2.2 Comparing Against AES

2.3 Using a Prebuilt Sequence

# 3. Core Components

3.1 MCW – Mango Cryptographic Workbench

3.2 MEC – Mango Encryption Core

3.3 MangoCipher – Mango Encryption Sample

3.4 Interaction Flow: MCW → MEC → MangoCipher

# 4. Transforms

4.1 What is a Transform?

4.2 Atomic vs Composite Transforms

4.3 Feedback & Coin System

# 5. Metrics & Evaluation

5.1 Metric Descriptions (Entropy, Avalanche, etc.)

5.2 PassCount & Grading

5.3 Scoring & Optimization

# 6. CutList & FailDB

6.1 Purpose of CutList

6.2 Purpose of FailDB

6.3 How They Guide Optimization

# 7. Sequence Analysis & Munges

7.1 What is a Munge?

7.2 Munge Levels (L1–L6)

7.3 Autotuning & Optimization Routines

7.4 Output Files (Contenders, State, Logs)

# 8. Integration

8.1 Using EPC in Your App

8.2 Customizing ACSCipher

8.3 Exporting Sequences from CRW

# 9. Advanced Topics

9.1 Coin Theory & Feedback Models

9.2 Input Entanglement Techniques

9.3 Sequence Reversibility & Transform Safety

# 10. Terms & Definitions

10.1 Mango Vocabulary

10.2 Sequence Format Reference

10.3 Common Abbreviations

# 10. Terms & Definitions

## 10.1 Mango Vocabulary

|  |  |
| --- | --- |
| Term | Definition |
| Transform | A modular operation applied to input data (e.g., byte shuffling, bit masking). Transforms are the atomic building blocks of Mango sequences. |
| Sequence | A specific ordered list of transforms. Sequences define how data is processed through Mango. |
| Munge | Mango’s automated process for discovering optimal transform sequences through exhaustive search and scoring. |
| Munge(A) | The baseline Munge process that tests transform orderings using a flat global round count (GR:9). |
| Munge(B) / BTR | Best-fit Transform Round optimization. Fine-tunes per-transform round counts (TR) for sequences identified by Munge(A). |
| .gs1/.gs2/.gs3/.gs4 | Mango sequence file formats. Represent various stages or types of sequence optimization (e.g., BTR-tuned, reordered, experimental). |
| Feedback (Fb) | A transform trait where the output is influenced by a running state (feedback), improving avalanche and entropy characteristics. |
| Coin | A session-specific, key-derived selector used in feedback-aware transforms to control variability and security. |
| DataHint | An input classification used to guide transform selection (e.g., Natural, Random, Combined). |
| CutList | A curated list of top-performing transforms for a given data type, used to filter future Munge runs. |
| FailDB | A database of previously rejected sequences that failed to meet minimum thresholds or had known weaknesses. |

## 10.2 Sequence Format Reference

|  |  |
| --- | --- |
| File Extension | Description |
| .gs1 | BTR-optimized sequences (transform rounds tuned individually). |
| .gs2 | Reordered + BTR sequences (BTR plus smart reshuffling). |
| .gs3 | Experimental or curated sequences, may include handcrafted candidates. |
| .gs4 | Smart sequences generated by advanced logic in Munge(S). |
| .me | Legacy format for `.gs3` files (experimental sequences). |
| .gse | Legacy format for `.gs2` files (BTR with enhanced ordering). |
| .txt | Log files and contender outputs. Often include pass count, mode, and data type metadata. |
| .json | Used for SaveMungeState, CutList cache, or MetricWeight configurations. |

## 10.3 Common Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| MCW | Mango Cryptographic Workbench |
| MEC | Mango Encryption Core |
| MangoCipher | Mango Encryption Sample |
| GR | Global Rounds – the number of times the full sequence is applied during encryption/decryption. |
| TR | Transform Rounds – the number of iterations applied within a single transform. |
| BTR | Best-fit Transform Round – the optimization routine that tunes TR per transform. |
| ACS | Adaptive Cipher Suite (legacy naming) |
| L1–L6 | Munge Levels – indicate how many transforms are used in a sequence (e.g., L4 = up to 4 transforms). |
| DN / DR / DS / DC | Data Type hints: Natural, Random, Sequence, Combined |
| ST / SF | Scoring mode flags: UseMetricScoring = True / False |
| PC# | Pass Count required for a sequence to be considered successful (e.g., PC6 = passed all 6 tests). |

# 9. Advanced Topics

## 9.1 Coin Theory & Feedback Models

Mango employs a dual-model approach to enhancing data diffusion: Coin-based input entanglement and feedback-aware transforms. Coins are input-derived, deterministic selectors generated from a hash of the input and IV, forming a 256-byte Coins table. Each transform is assigned a unique Coin index, ensuring behavior tailored to the input while remaining stateless and reversible. Meanwhile, feedback transforms such as MaskedCascadeSubFbTx introduce additional avalanche effects by using dynamic pseudo-random state within each transform iteration, while still preserving invertibility.

## 9.2 Input Entanglement Techniques

Input Entanglement in Mango is primarily achieved through the use of Coins and CBox systems. Coins inject deterministic variability into each transform by modifying behavior based on input hashes, while the CBox is a shuffled substitution table derived from the password or seed. Transforms like FrequencyEqualizerTx, SubBytesXorMaskTx, and BitFlipCascadeTx utilize these mechanisms to tightly bind output to both the input and session state, making unauthorized decoding extremely difficult.

## 9.3 Sequence Reversibility & Transform Safety

Each transform in Mango explicitly declares its inverse, enabling full bidirectional encryption and decryption. Transforms must pass a Validator check to ensure preconditions for safe use are met (e.g., balance > 0 for inverse). Mango also assigns CoinPreferences in synchronized pairs to ensure that forward and inverse transforms access matching selectors. The design guarantees that all transforms are reversible when used correctly, preventing data loss or corruption.