TREND MICRO

Securing Your Journey
to the Cloud

# A Theoretic Framework for Evaluating Similarity Digesting Tools

**LIWEI REN, PH.D**, Trend Micro™

DFRWS EU 2015, Dublin, Ireland, March, 2015

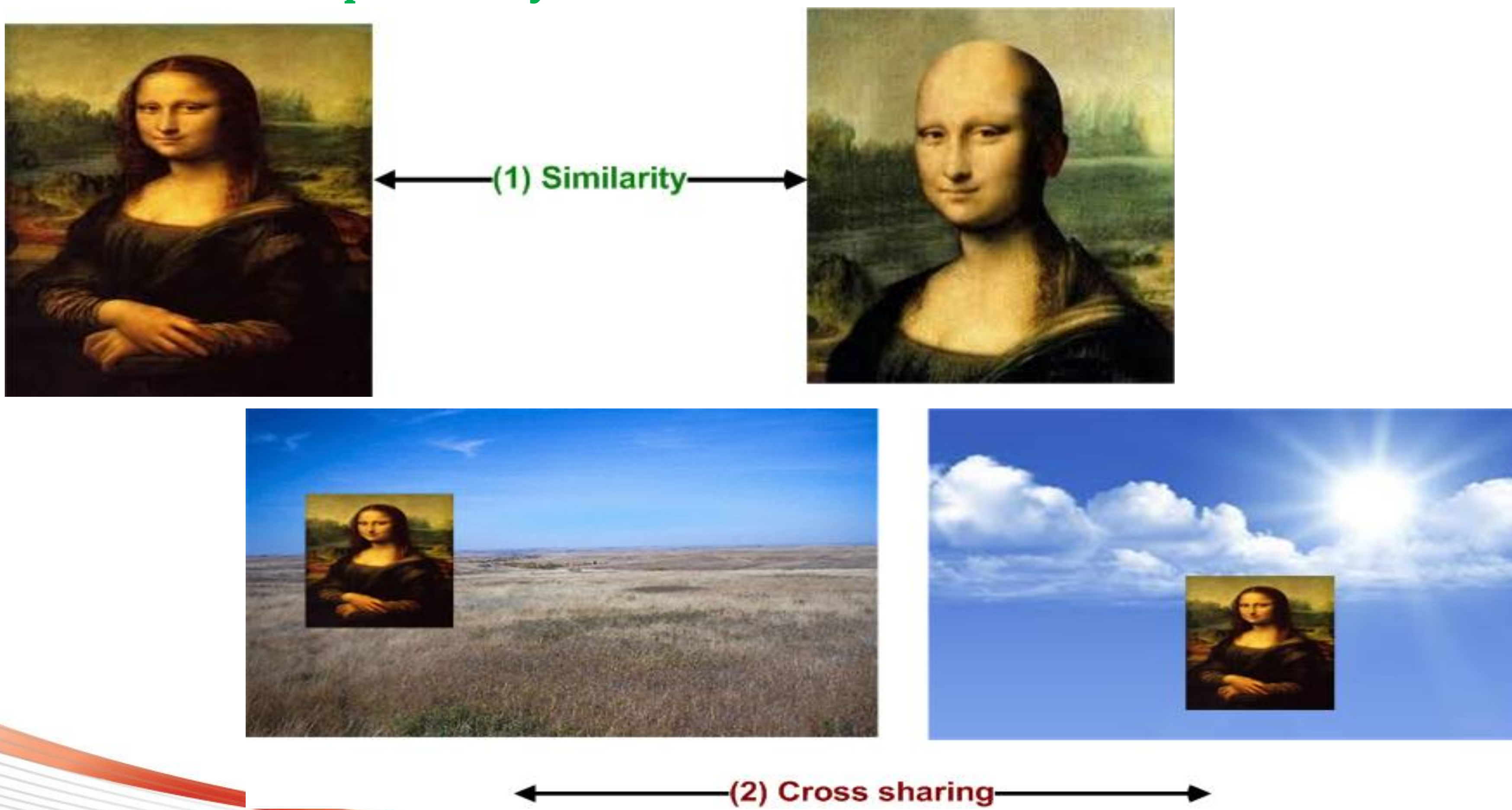
 

# Agenda

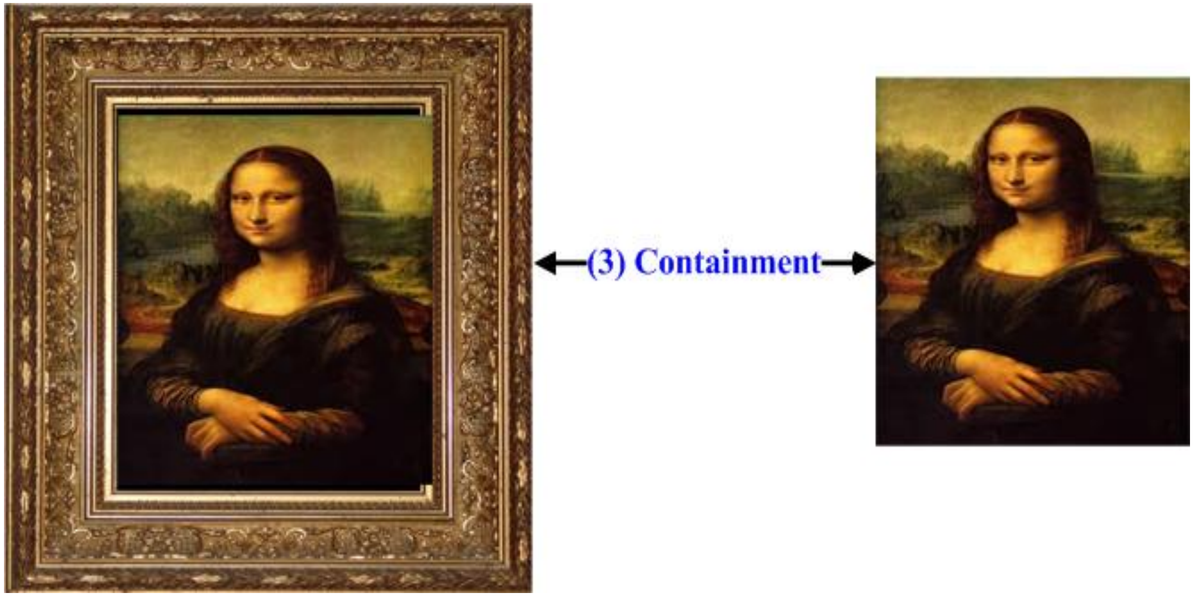
- Byte-wise Approximate Matching
- Similarity Digesting Tools
- Mathematical Models for Byte-wise Similarity
- Tool Evaluation with Theoretic Analysis
- Tool Evaluation with Data Experiment
- Further Research for Approximate Matching

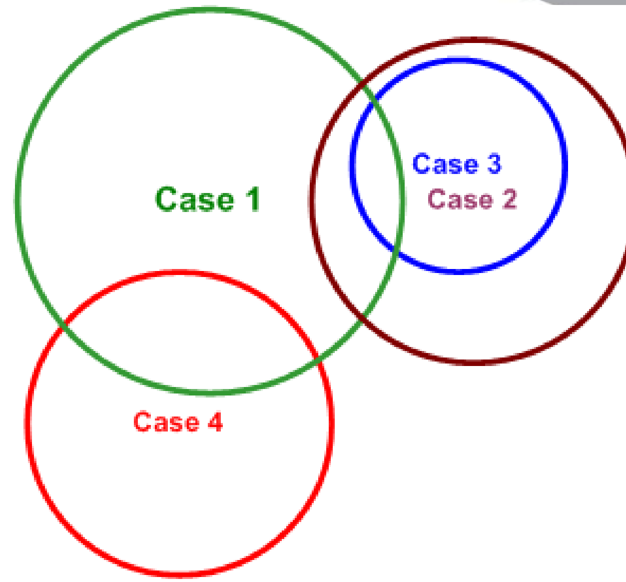
# Byte-wise Approximate Matching

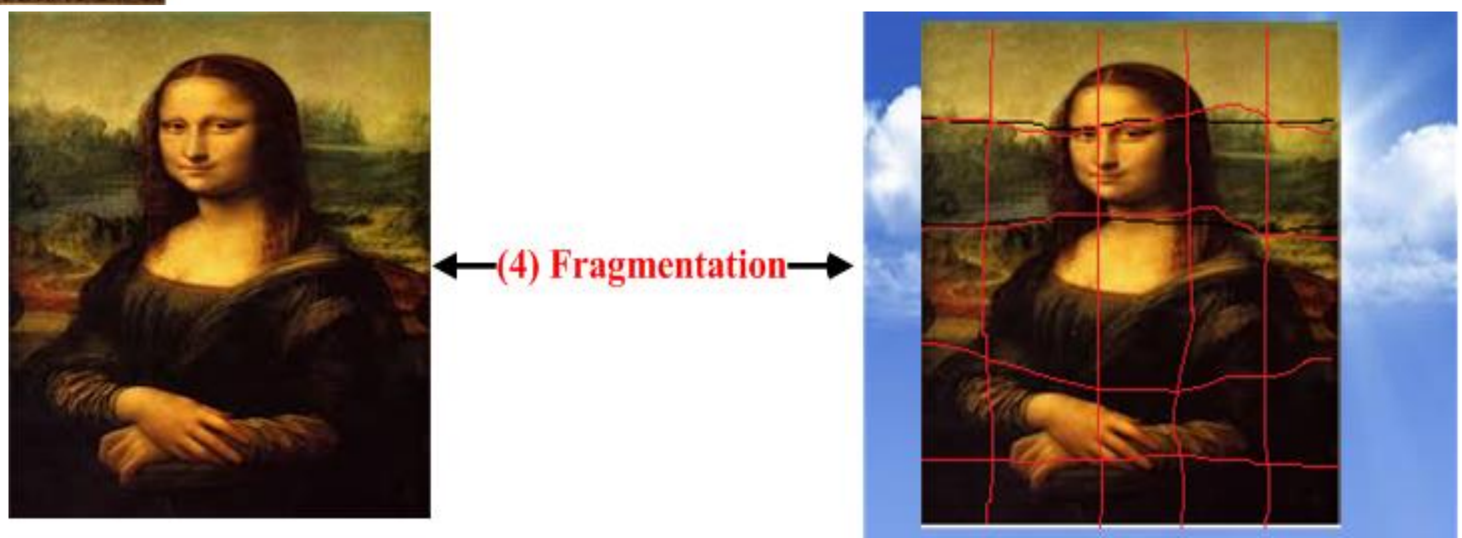
- Byte-wise similarity & approximate matching.
  - What is byte-wise similarity ?
- 4 Use Cases specified by *NIST*:



# Byte-wise Approximate Matching



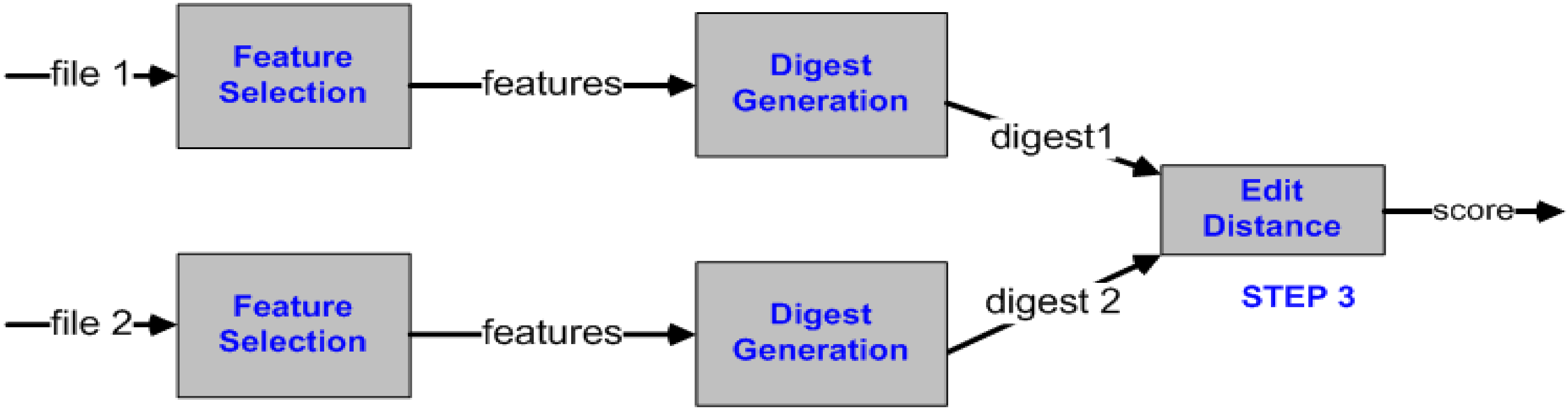



# Similarity Digesting Tools

- **Similarity digesting :**
  - A class of hash techniques or tools that preserve similarity.
  - Typical steps for digest generation:


  - Detecting similarity with similarity digesting:

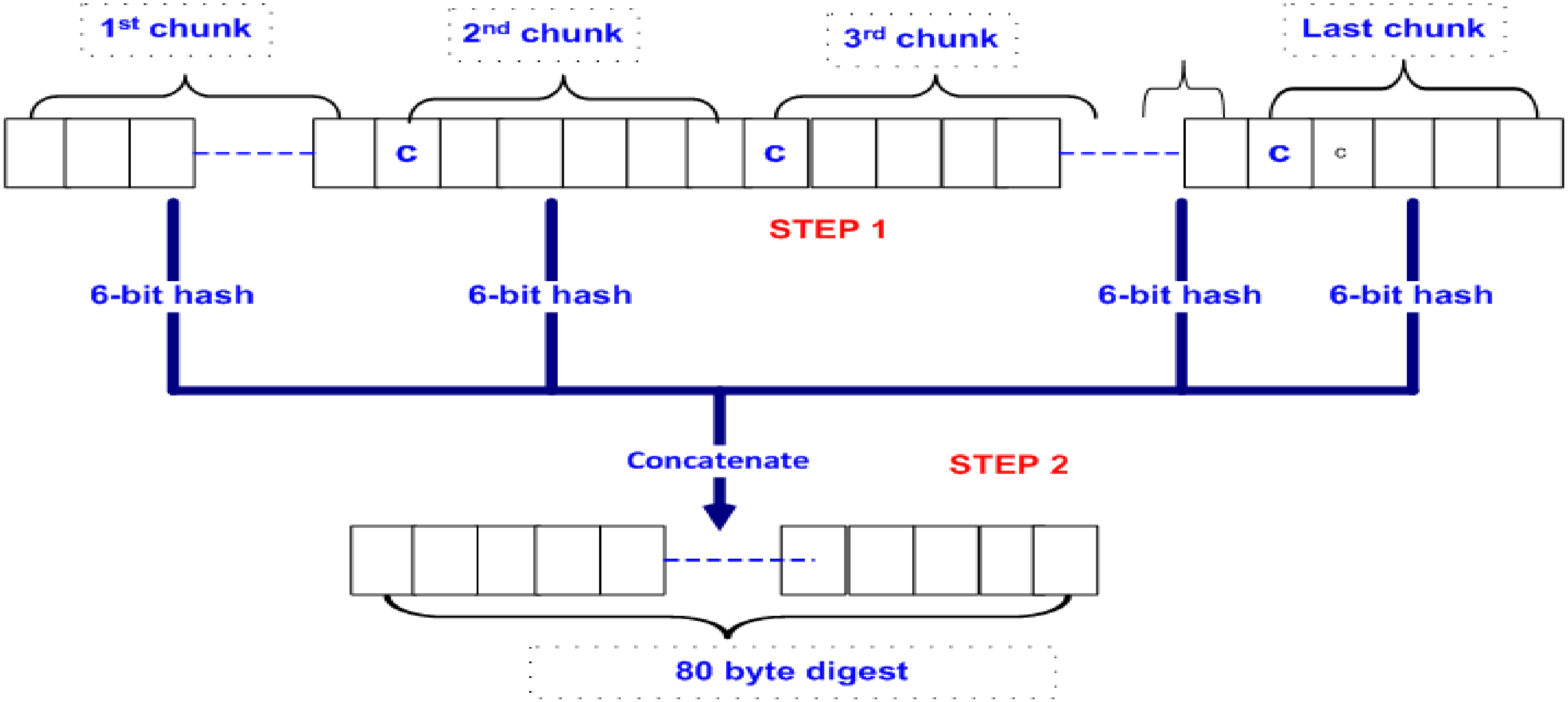



- **Three similarity digesting algorithms and tools:**
  - ssdeep, sdhash & TLSH

# Similarity Digesting Tools

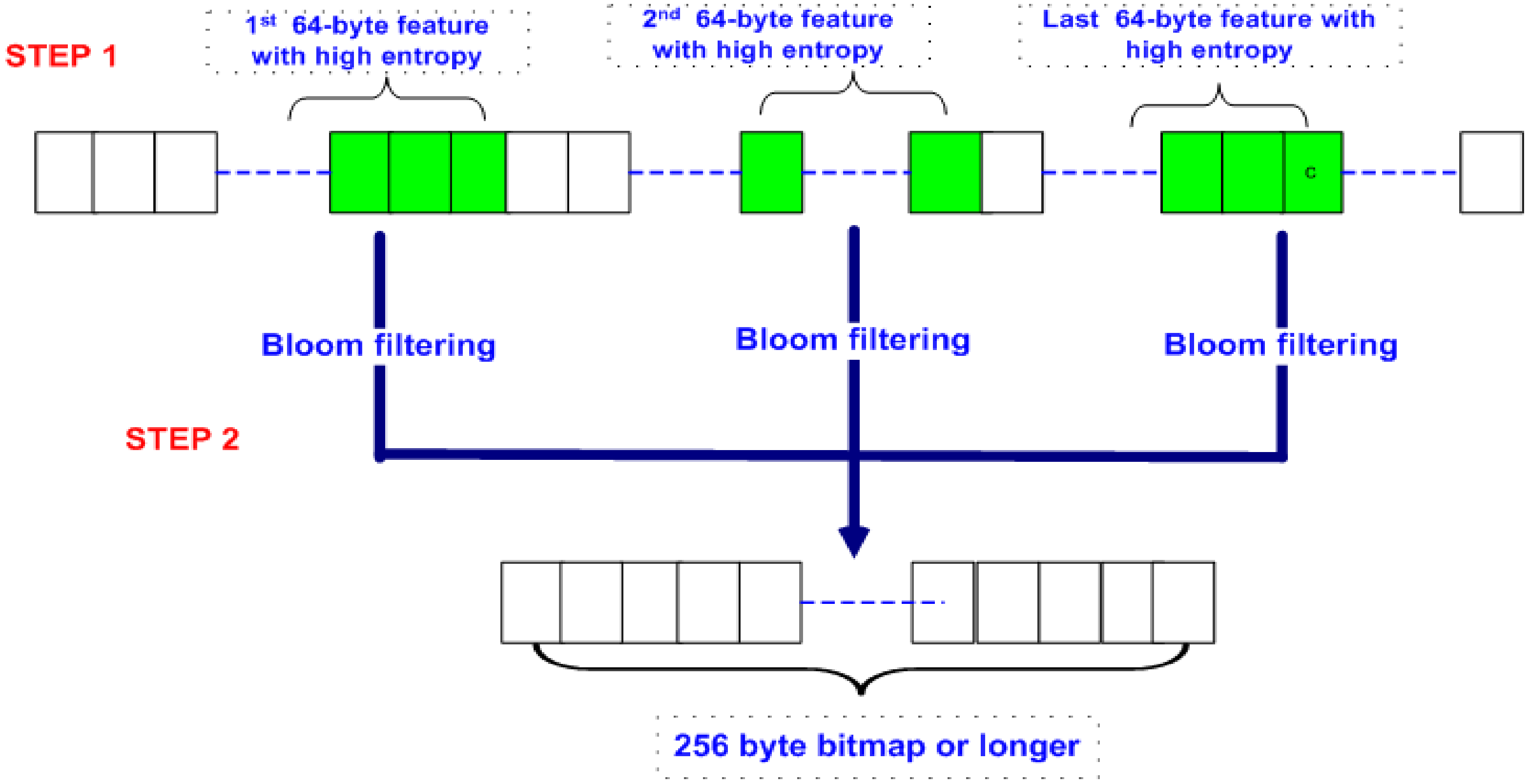
- ssdeep
  - Two steps for digesting:




  - **Edit Distance:** Levenshtein distance

# Similarity Digesting Tools

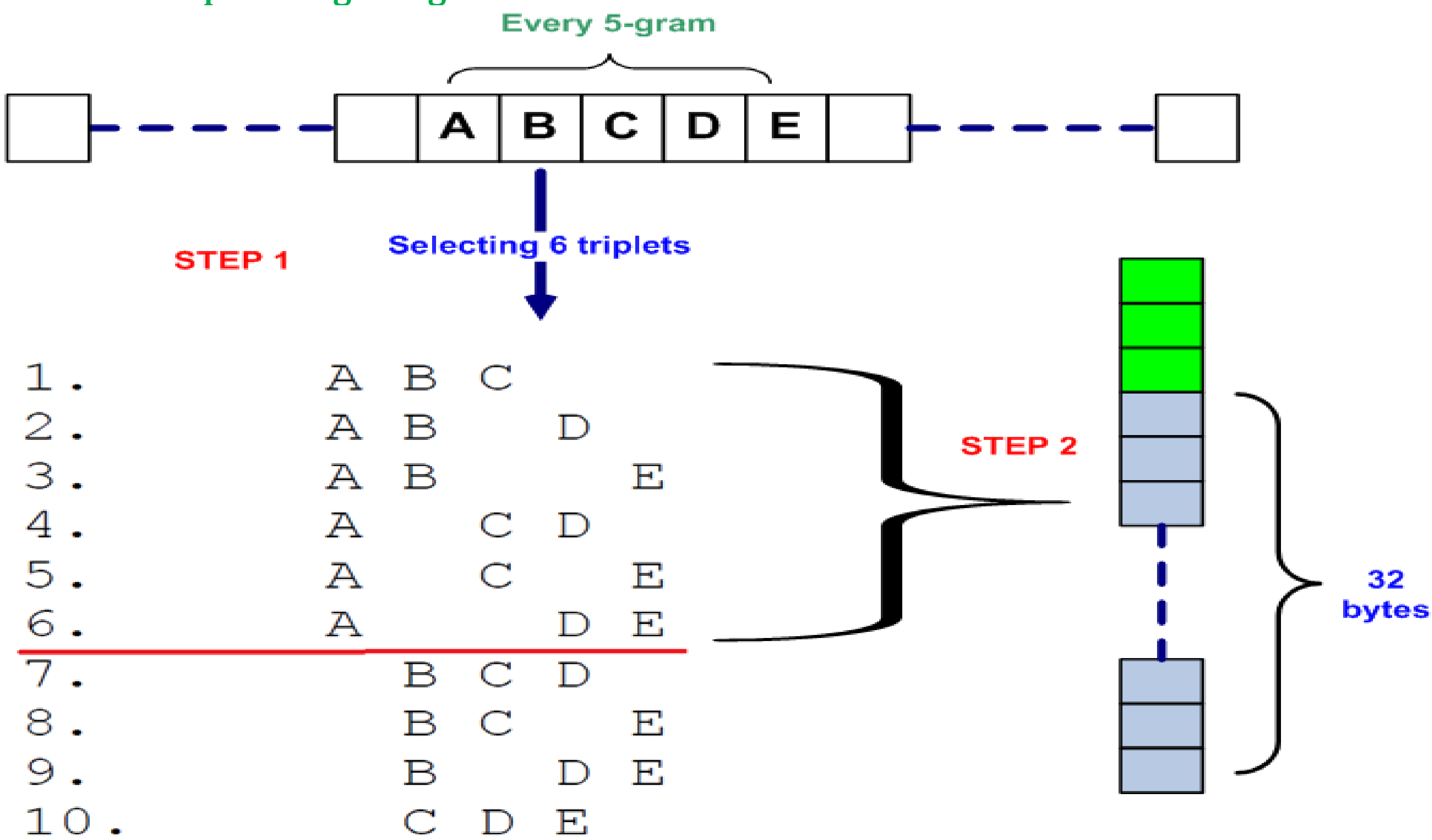
- sdhash
  - Two steps for digesting:




  - Edit Distance: Hamming distance

# Similarity Digesting Tools

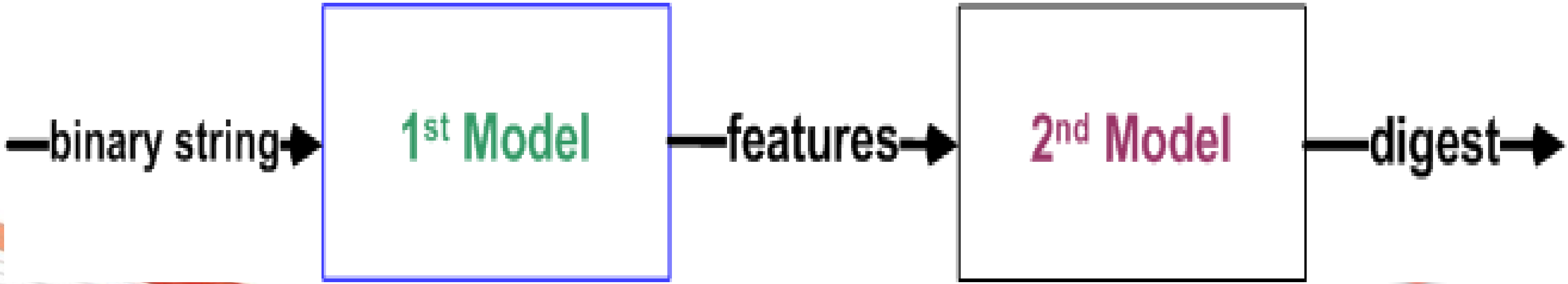
- **TLSH**
  - Two steps for digesting :




  - Edit Distance: A diff based evaluation function

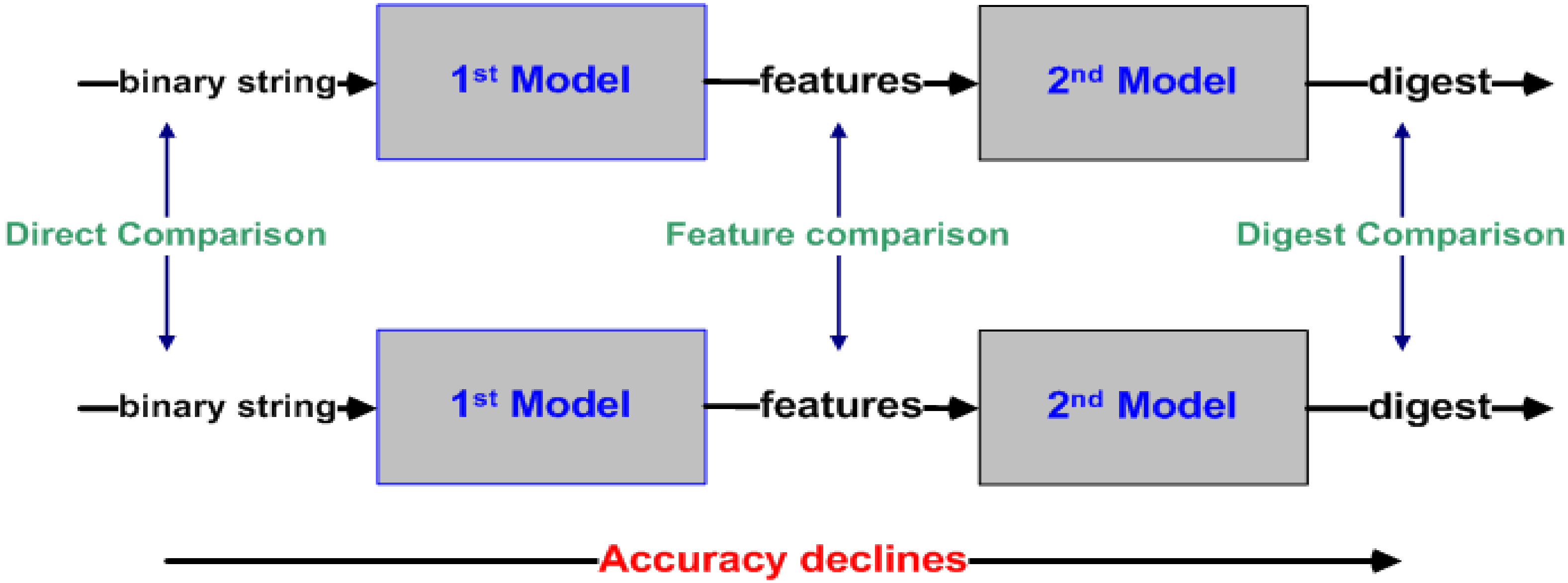
# Mathematical Models for Byte-wise Similarity

- **Summary of Three Similarity Digesting Schemes:**
  - Using a **first model** to describe a binary string with ***selected features***:
    - ssdeep model: a string is a ***sequence*** of ***chunks*** (split from the string).
    - sdhash model: a string is a ***bag*** of ***64-byte blocks*** (selected with entropy values).
    - TLSH model: a string is a ***bag*** of ***triplets*** (selected from all 5-grams).
  - Using a **second model** to map the selected features into a digest which is able to preserve similarity to certain degree.
    - ssdeep model: a sequence of chunks is mapped into a 80-byte digest.
    - **sdhash model**: a bag of blocks is mapped into one or multiple 256-byte bloom filter bitmaps.
    - TLSH model: a bag of triplets is mapped into a 32-byte container.



# Mathematical Models for Byte-wise Similarity

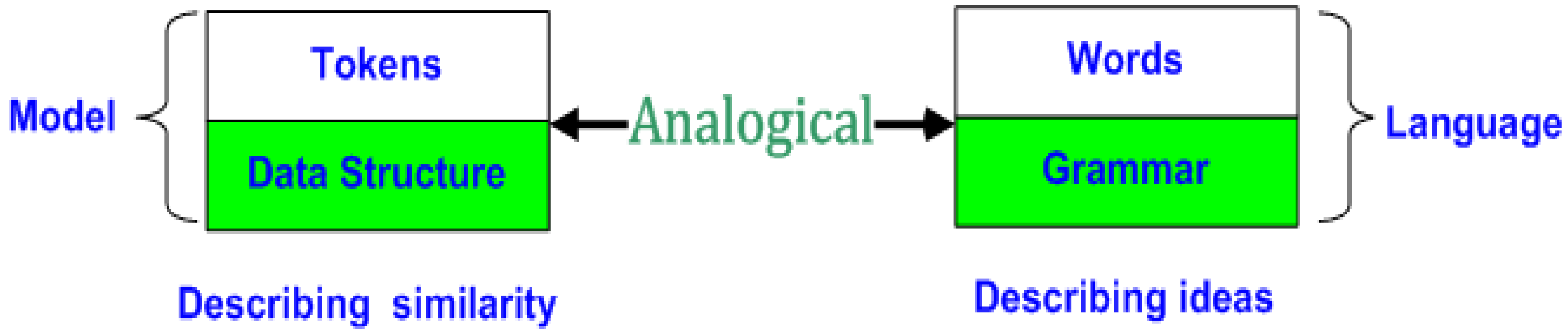
- Three approaches for similarity evaluation:




- 1st model plays critical role for similarity comparison.
  - Let focus on discussing various 1st models today.
    - Based on a unified format.
- 2nd model saves space but further reduces accuracy.

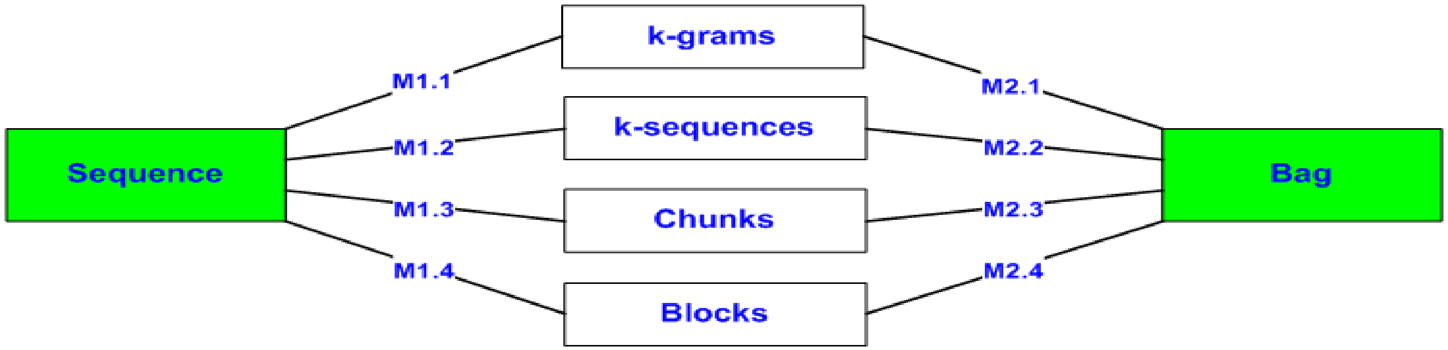
# Mathematical Models for Byte-wise Similarity

- **Unified format for 1st model:**
  - A string is described as a collection of **tokens** (aka, features) organized by a **data structure:**
    - ssdeep: a sequence of chunks.
    - sdhash: a bag of 64-byte blocks with high entropy values.
    - TLSH: a bag of selected triplets.
  - Two types of data structures: sequence, bag.
  - Three types of tokens: chunks, blocks, triplets.
- **Analogical comparison:**



# Mathematical Models for Byte-wise Similarity

- **Four general types of tokens from binary strings:**
  - k-grams where k is as small as 3,4,…
  - k-subsequences: any subsequence with length k. The triplet in TLSH is an example.
  - Chunks: whole string is split into non-overlapping chunks.
  - Blocks: selected substrings of fixed length.
- Eight different models to describe a string for similarity.




- Analogical thinking:
  - we define different distances to describe a metric space.

# Tool Evaluation with Theoretic Analysis

- **Data Structure:**
  - **Bag**: a bag ignores the order of tokens. It is *good at handling content swapping.*
  - **Sequence**: a sequence organizes tokens in an order. This is *weak for handling content swapping.*
- **Tokens**:
  - **k-grams**: Due to the small k ( 3,4,5,…), this fine granularity is *good at handling fragmentation*.
  - **k-sequences**: Due to the small k ( 3,4,5,…), this fine granularity is *good at handling fragmentation* .
  - **Chunks**: This approach takes account of every byte in raw granularity. It should be *OK at handling containment* and *cross sharing*
  - **Blocks**: Depending on different selection functions, even though it does not take account of every byte, but it may present a string more efficiently and that is good for generating similarity digests. Due to the nature of fixed length blocks, it is *good at handling containment* and *cross sharing*.

# Tool Evaluation with Theoretic Analysis

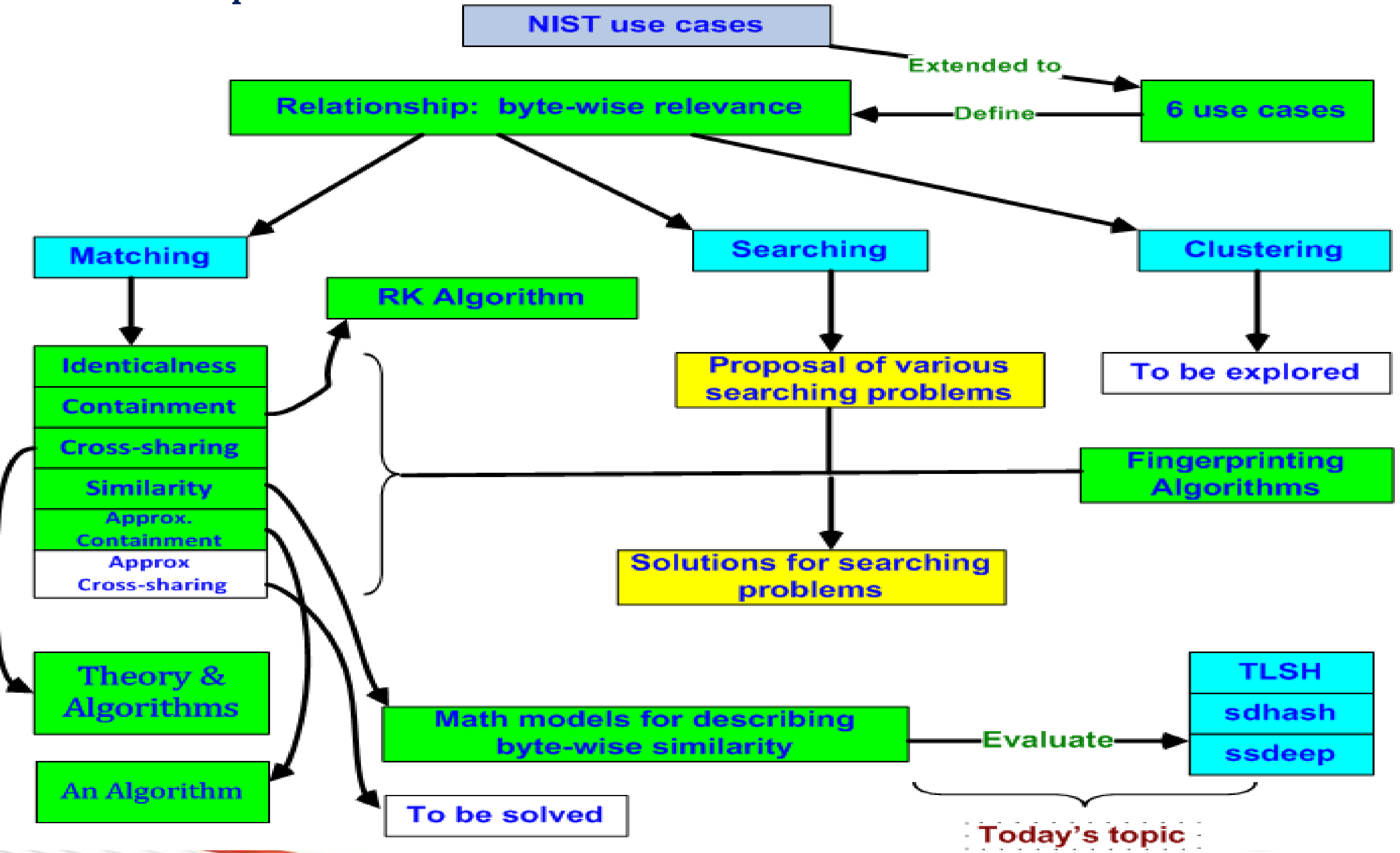
| Tool | Model | Minor Changes | Containment | Cross sharing | Swap | Fragmentation |
|---|---|---|---|---|---|---|
| ssdeep | M1.3 | ***High*** | ***Medium*** | ***Medium*** | ***Medium*** | ***Low*** |
| sdhash | M2.4 | ***High*** | ***High*** | ***High*** | ***High*** | ***Low*** |
| TLSH | M2.2 | ***High*** | ***Low*** | ***Medium*** | ***High*** | ***High*** |
| | | | | | | |
| Sdhash + TLSH | Hybrid | ***High*** | ***High*** | ***High*** | ***High*** | ***High*** |

# Tool Evaluation with Data Experiment

| | | Base File Size = 2MB | | | Base File Size = 64KB | | |
|---|---|---|---|---|---|---|---|
| Purpose of Tests | **Edit Operations** | **ssdeep** | **sdhash** | **TLSH** | **ssdeep** | **sdhash** | **TLSH** |
| **Containment** | Cut 30% at the beginning | 82 | 60 | 31 | 79 | 89 | 31 |
| | Cut 60% at the end | 54 | 100 | X | 58 | 99 | X |
| | Cut 90% at the beginning | X | 77 | X | X | 100 | X |
| **Cross sharing** | Substitute 30% at the end | 72 | 70 | 69 | 75 | 59 | 68 |
| | Substitute 60% in the end | 47 | 40 | 54 | 47 | 37 | 62 |
| | Substitute 90% at the end | 29 | 10 | 47 | X | 6 | 42 |
| **Swap** | Swap with 2-1 | 52 | 71 | 99 | 54 | 68 | 98 |
| | Swap with 4-3-2-1 | 36 | 59 | 98 | 33 | 54 | 98 |
| | Swap with 8-7-6-5-4-3-2-1 | 32 | 62 | 99 | X | 48 | 96 |
| **Fragmentation** | Modify the bytes at 64*j | X | X | 58 | X | X | 78 |
| | Modify the bytes at 128*j | X | X | 78 | X | X | 83 |
| | Modify the bytes at 256*j | X | 15 | 86 | X | 33 | 82 |
| **Minor changes** | Swap with 1-2-3-4-5-7-6-8-9-10-11-12-13-14-15-16. Subst 1% at the end. Cut 1% at the beginning. | 90 | 88 | 93 | 83 | 93 | 84 |
| | Swap with 1-2-3-5-4-6-7-8-9-10-11-12-13-14-15-16. Cut 2% in the beginning. Subst 1% at the end. | 91 | 85 | 92 | 82 | 82 | 86 |

# Further Research for Approximate Matching

- A Roadmap for Further Research :



# Q&A

- Thank you for your interest.
- Any questions?
- My Contact Information:
  - Email: liwei_ren@trendmicro.com
  - Linkedin: https://www.linkedin.com/in/drliweiren
  - Academic Page: https://pitt.academia.edu/LiweiRen