M^2R: Enabling Stronger Privacy in MapReduce Computation

Anh Dinh, Prateek Saxena,

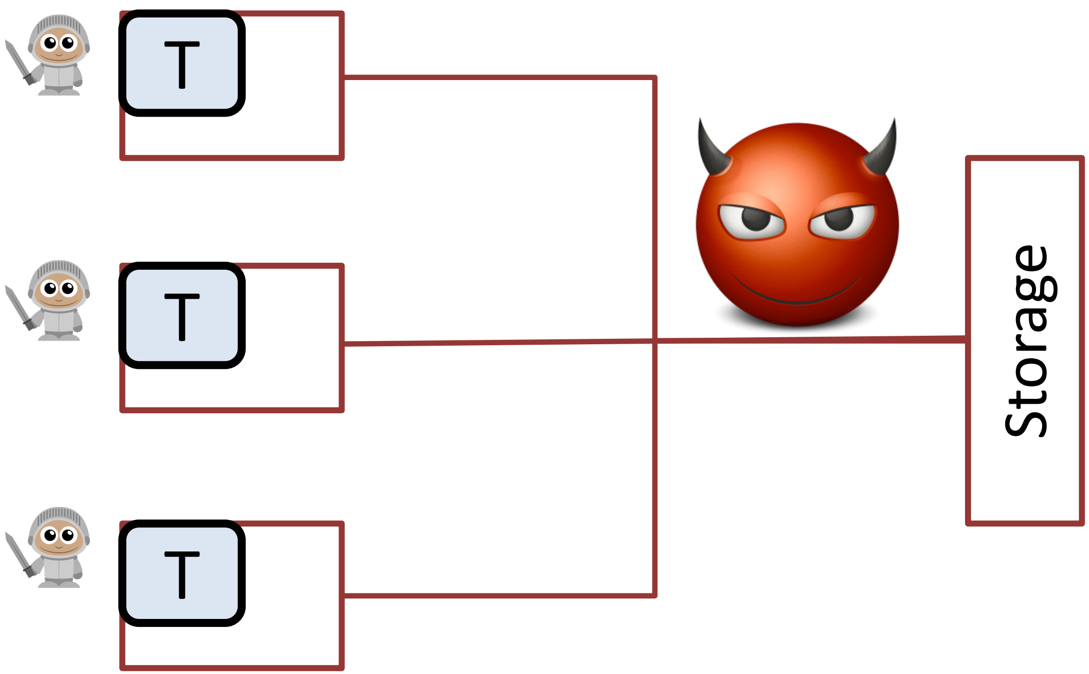
<u>Ee-Chien Chang</u>, Beng Chin Ooi, Chunwang Zhang

School of Computing

National University of Singapore

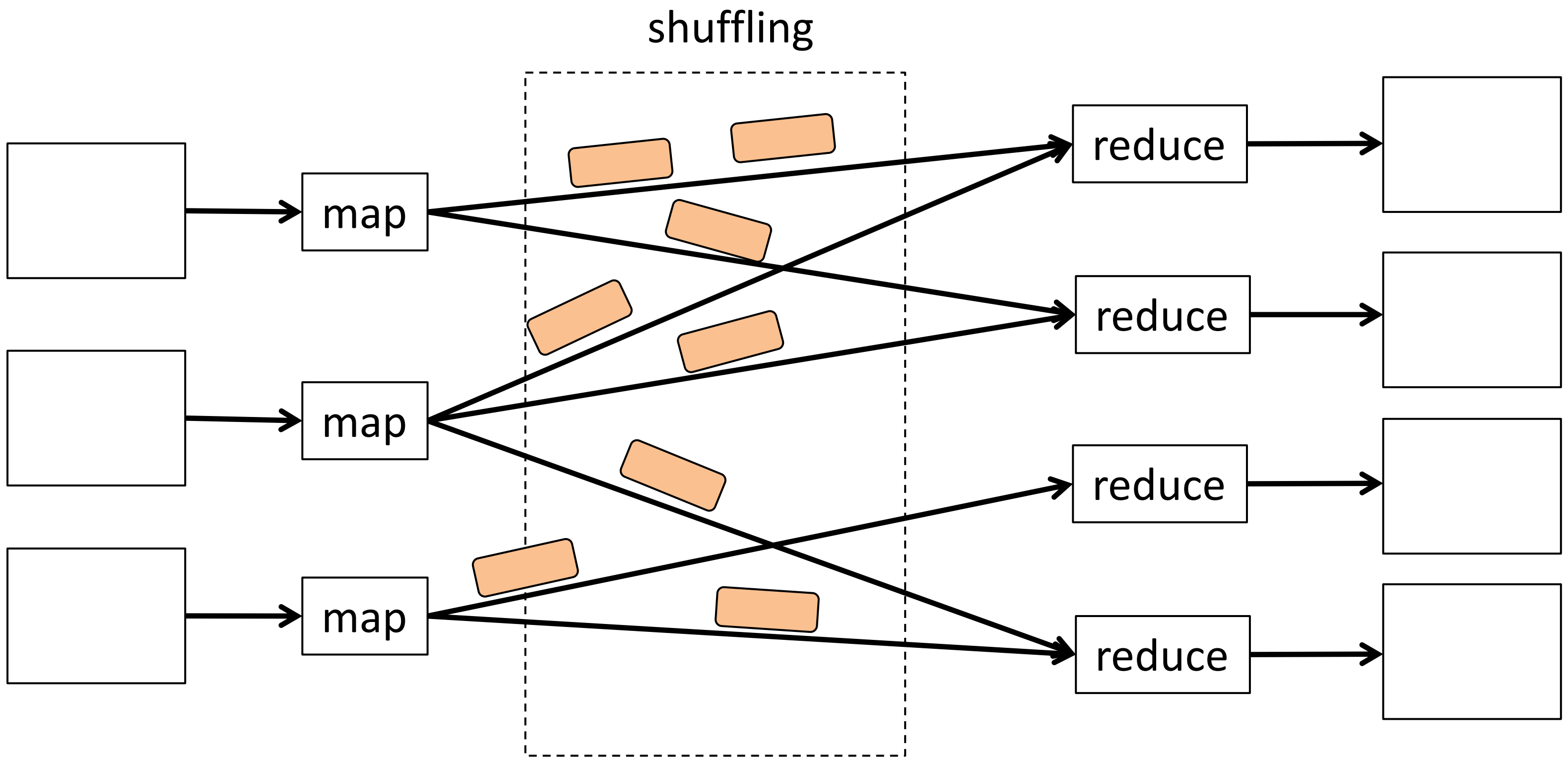
1. Motivation

- Distributed computation (MapReduce) on large dataset with Trusted computing.



- Integrity + Confidentiality.
- Applicable in private or public cloud setting.

Background: MapReduce



- Computation & "shuffling" of <key, value> tuples.
- Phases: Map → Shuffle → Reduce.
- "map" outputs a set of tuples.
- During Shuffling, tuples are grouped according to their key.
- Each "reduce" instance corresponds to an unique key **k**. It takes all tuples with the key **k** and output a set of tuples.

Background: Hadoop

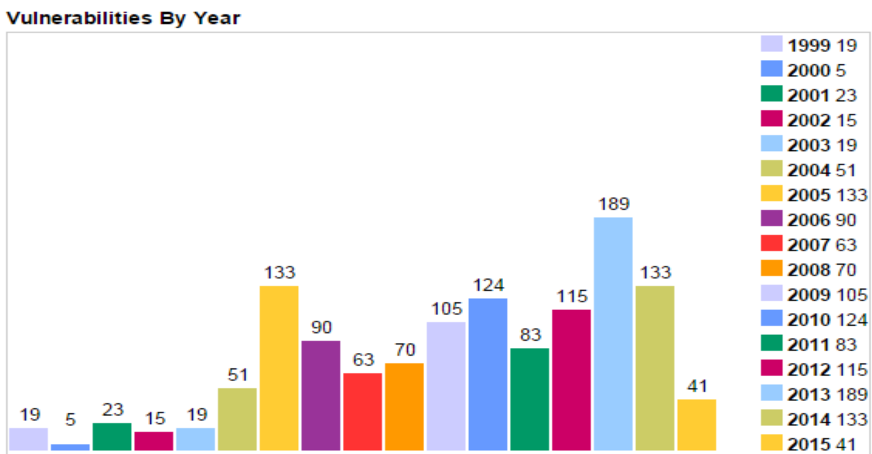
- Hadoop: software framework written in Java
- ≈ 190K LOC (Hadoop 0.21.0)
- Consists of MapReduce modules, Hadoop Distributed File System (HDFS), etc.

Challenge 1: Keep Trusted Code Base small

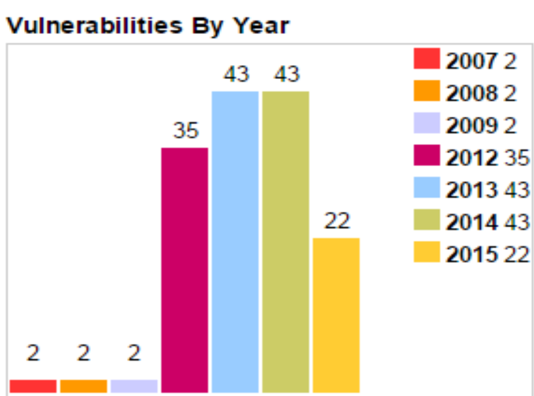
Application Frameworks

Operating Systems

Hypervisor

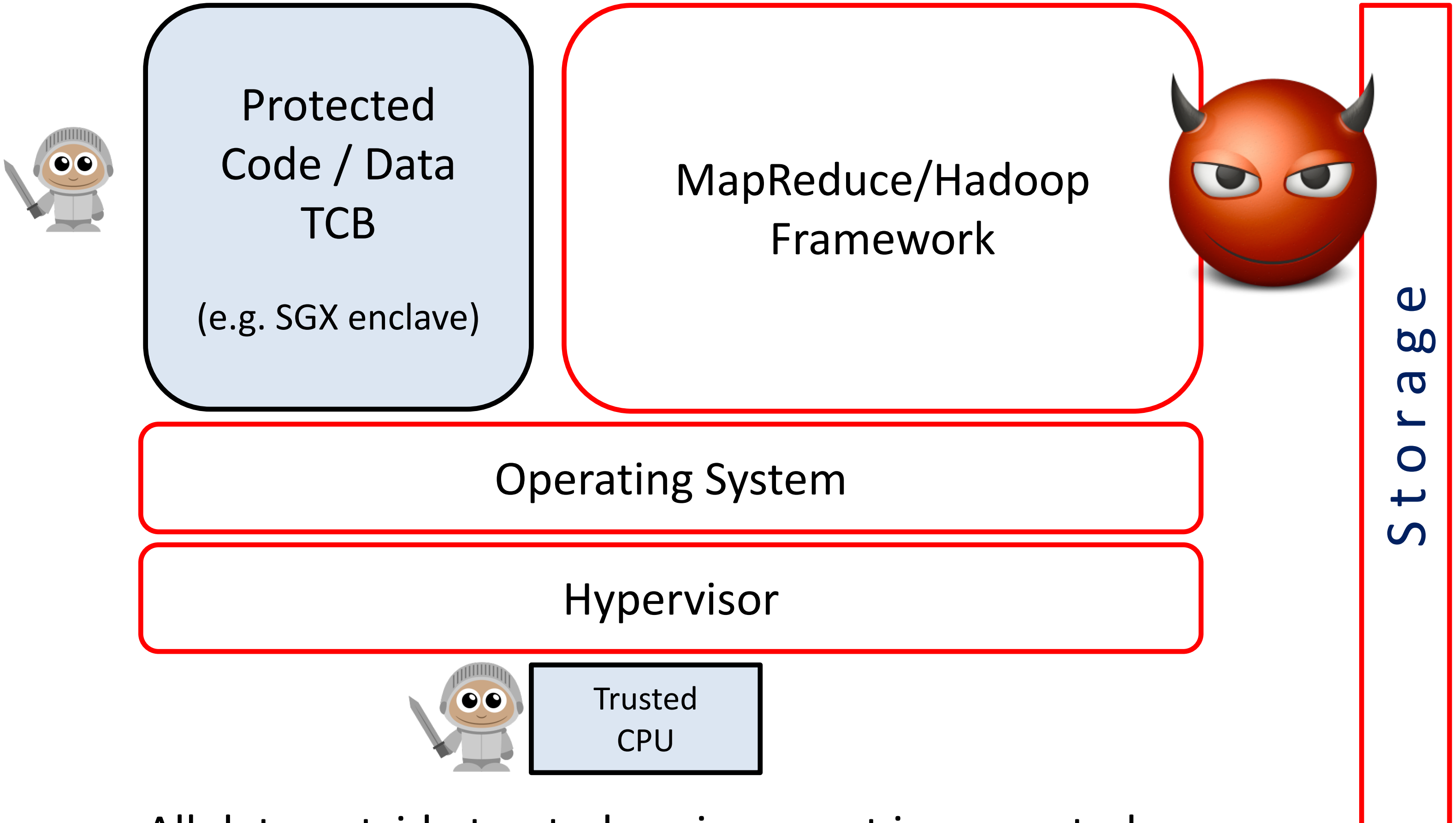


CVEs in Linux [CVE-DB]



Affected many hypervisors (e.g Xen / KVM) [CS Report]

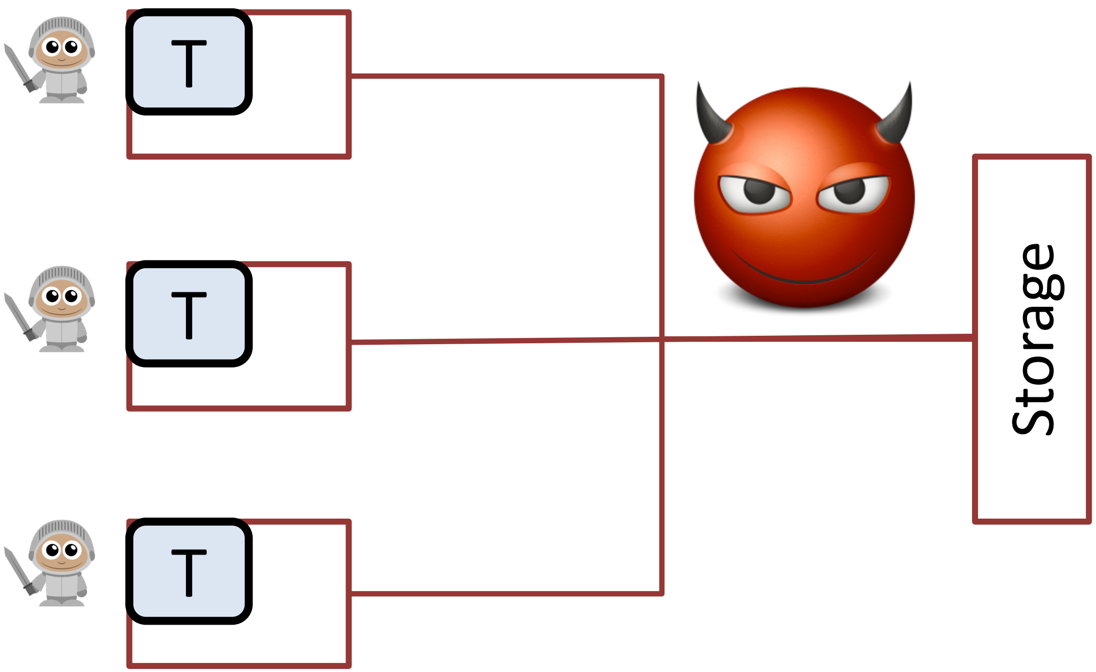
Challenge 1: Keep TCB small



- All data outside trusted environment is encrypted
- Software-only attack.

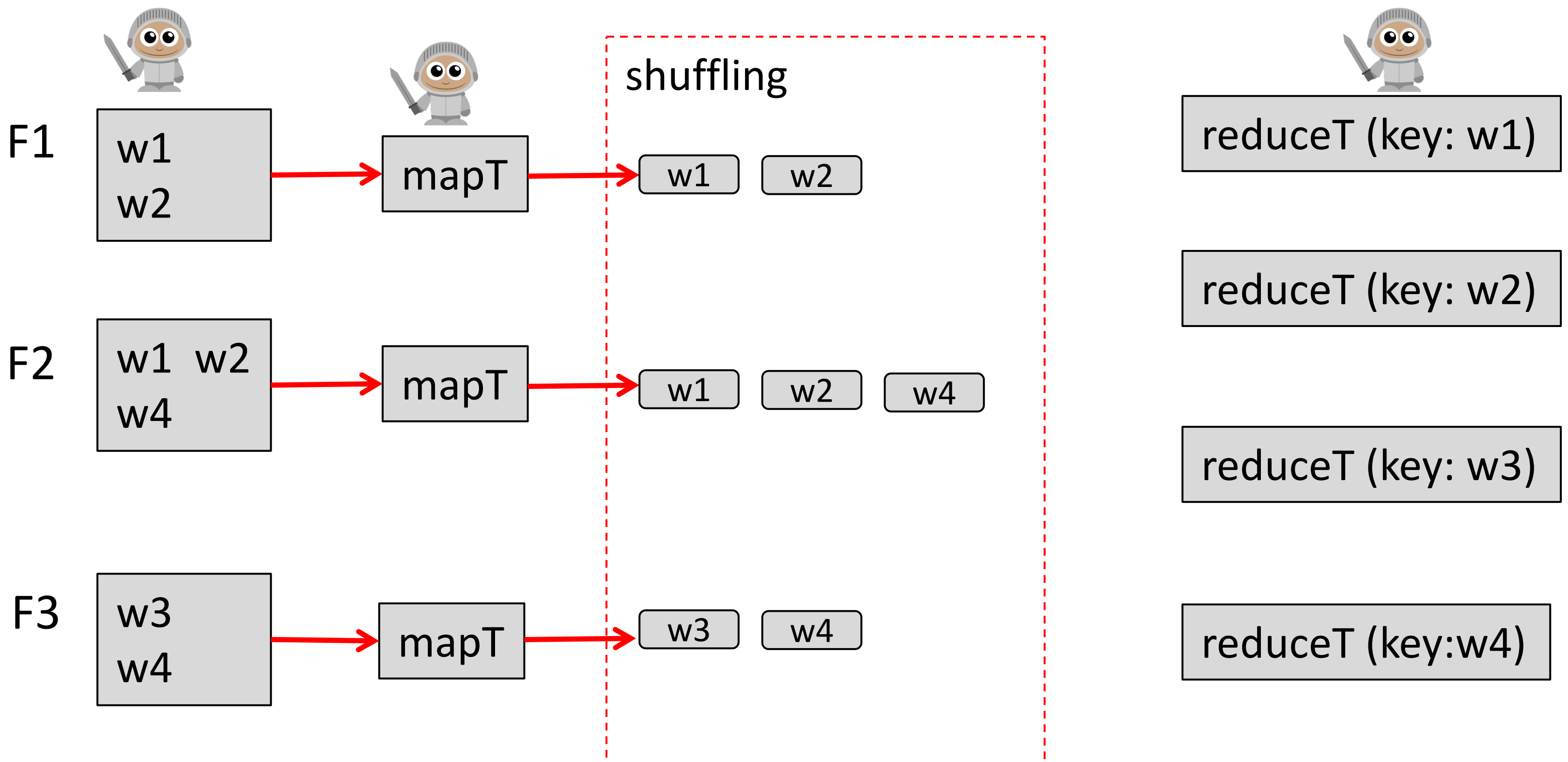
Identify small essential components of MapReduce to be included in the TCB.

Challenge 2: Interactions Leaks Info



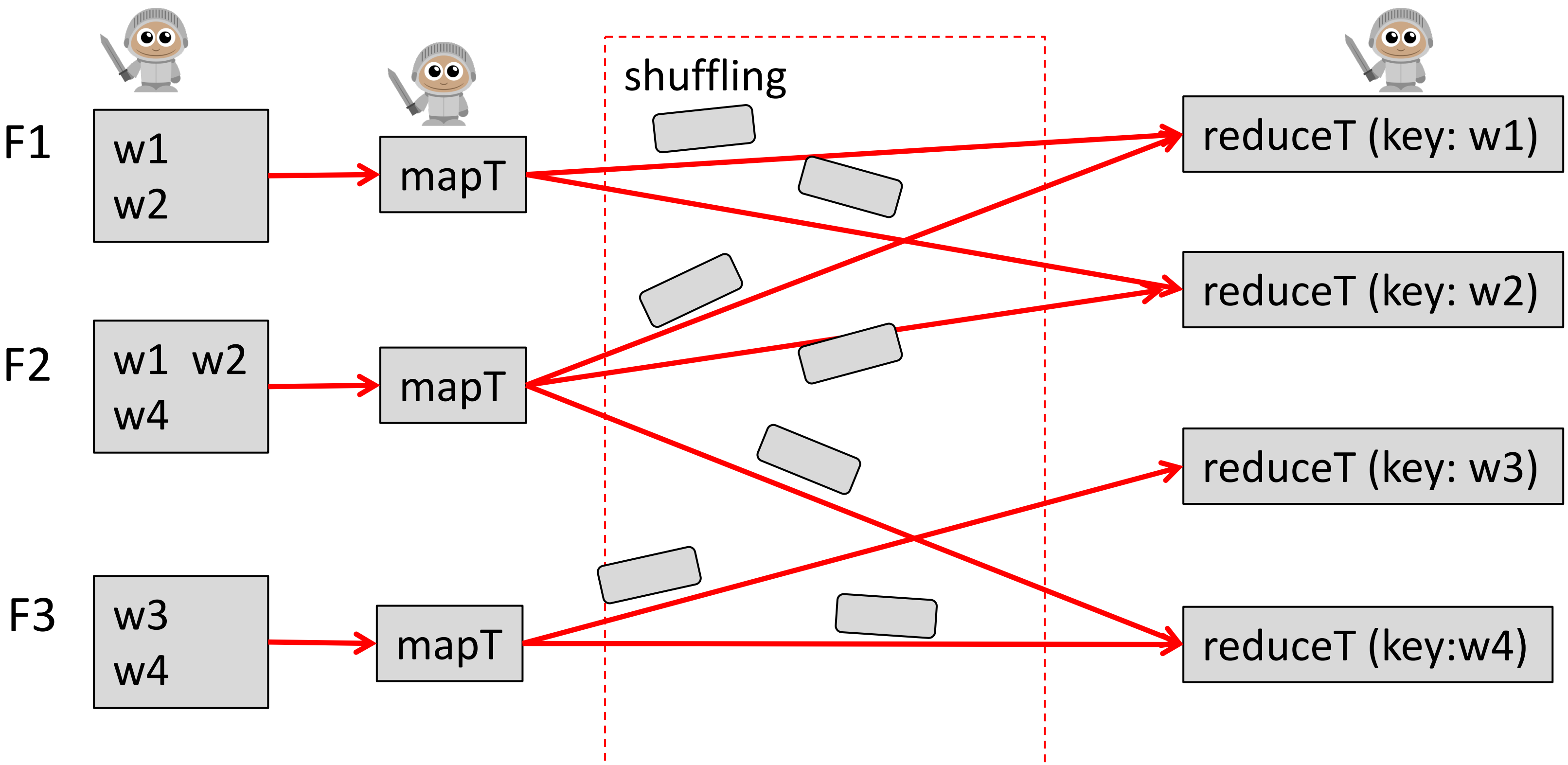
Example of leakage: wordcount

- Map Phase: each mapT generates the tuples.



Example of leakage: wordcount

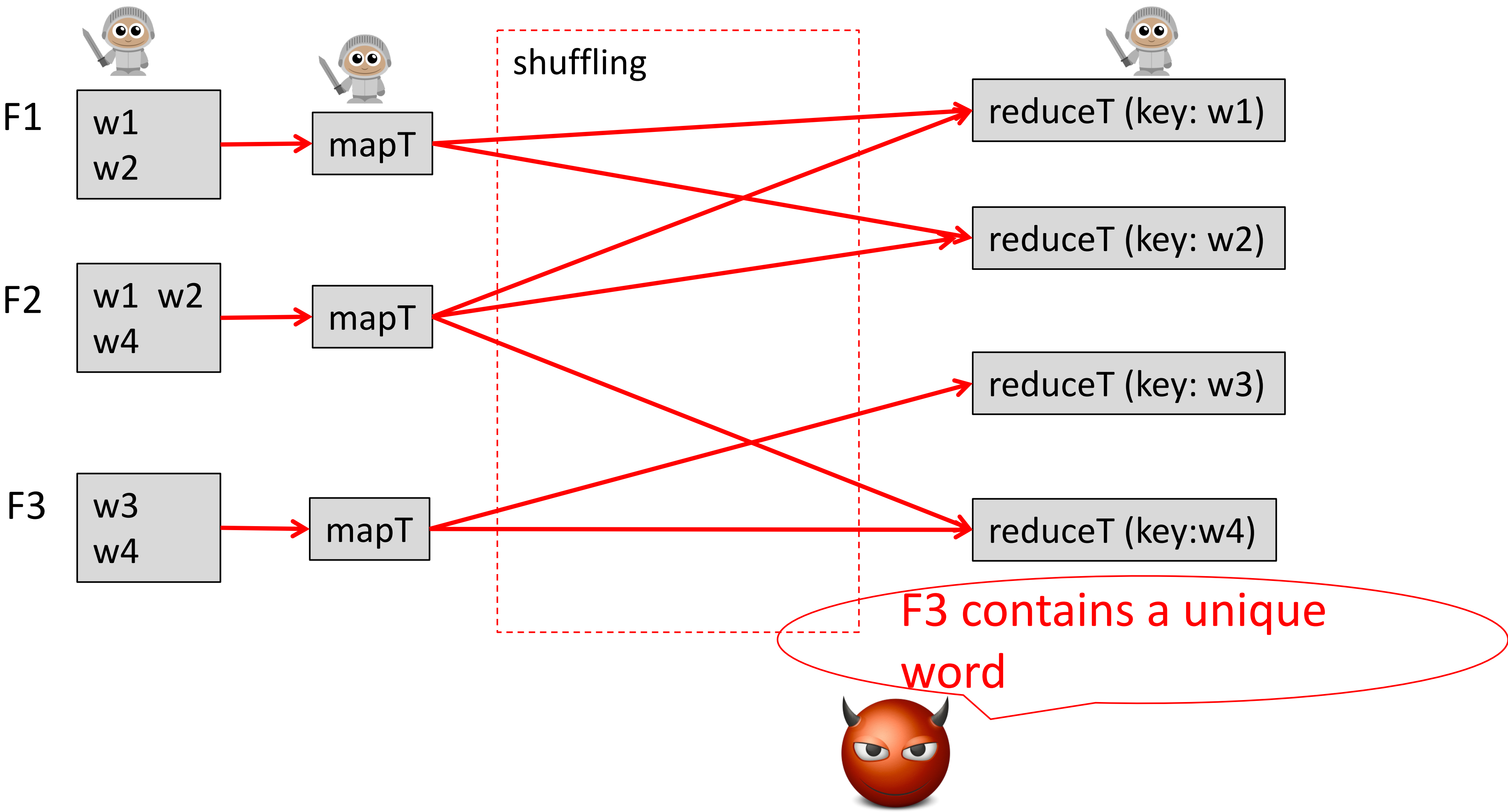
- Shuffling Phase: The tuples are grouped w.r.t the “words”.



- Reduce Phase: reduceT counts and outputs the number of tuples it received.

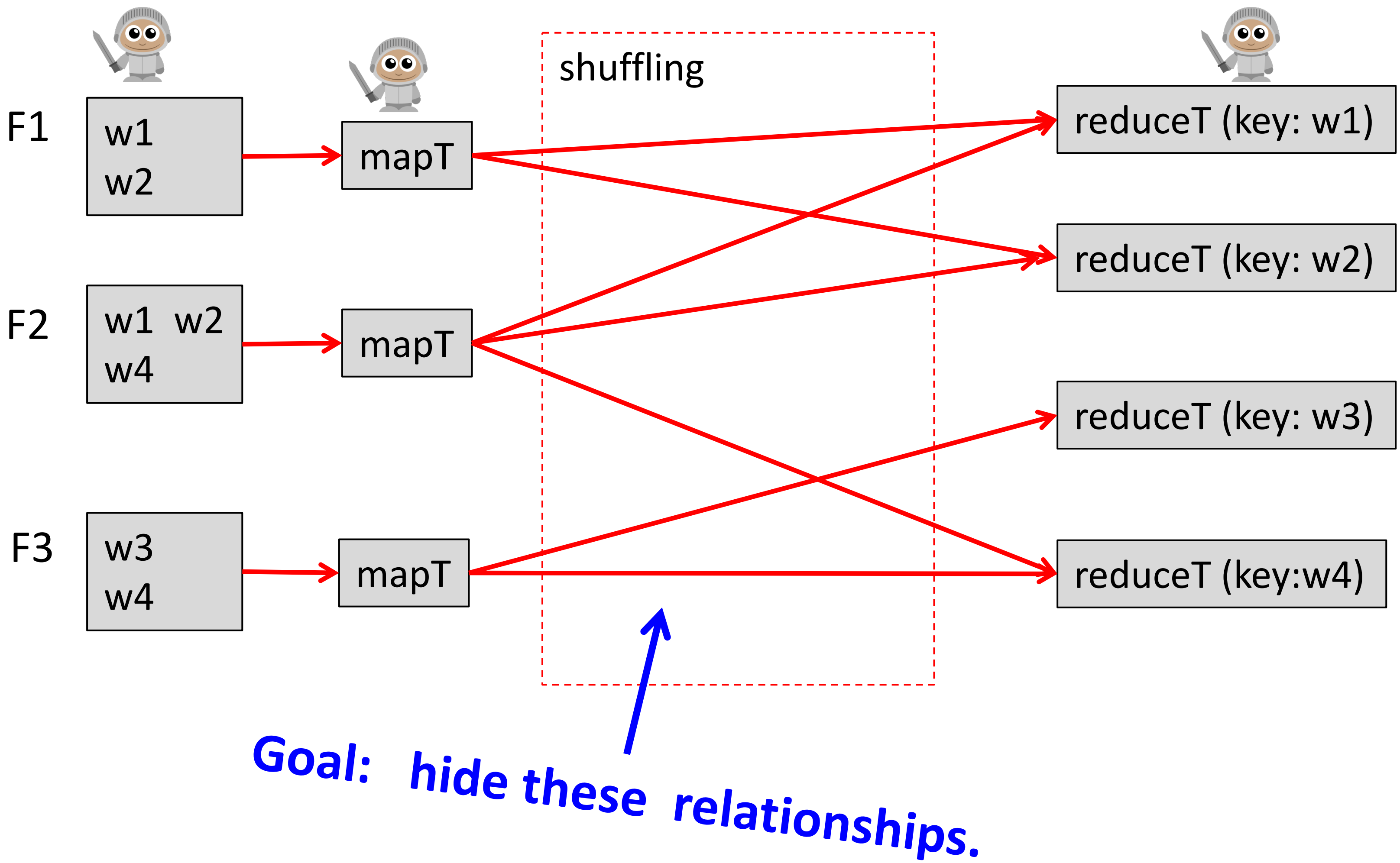
Example of leakage: word counts

- By observing the flow of tuples, one can infer relationships among the input files.



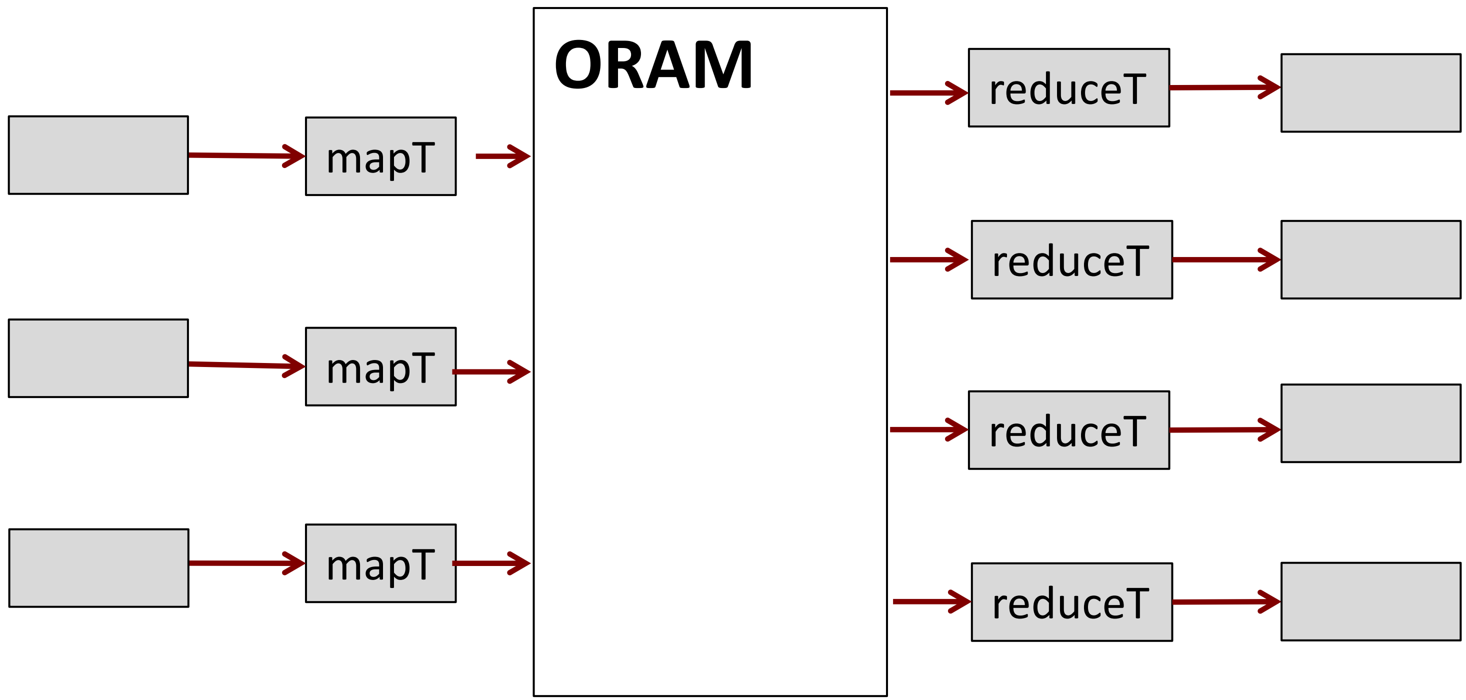
Example of leakage: word counts

- By observing the flow of tuples, one can infer relationships among the input files.



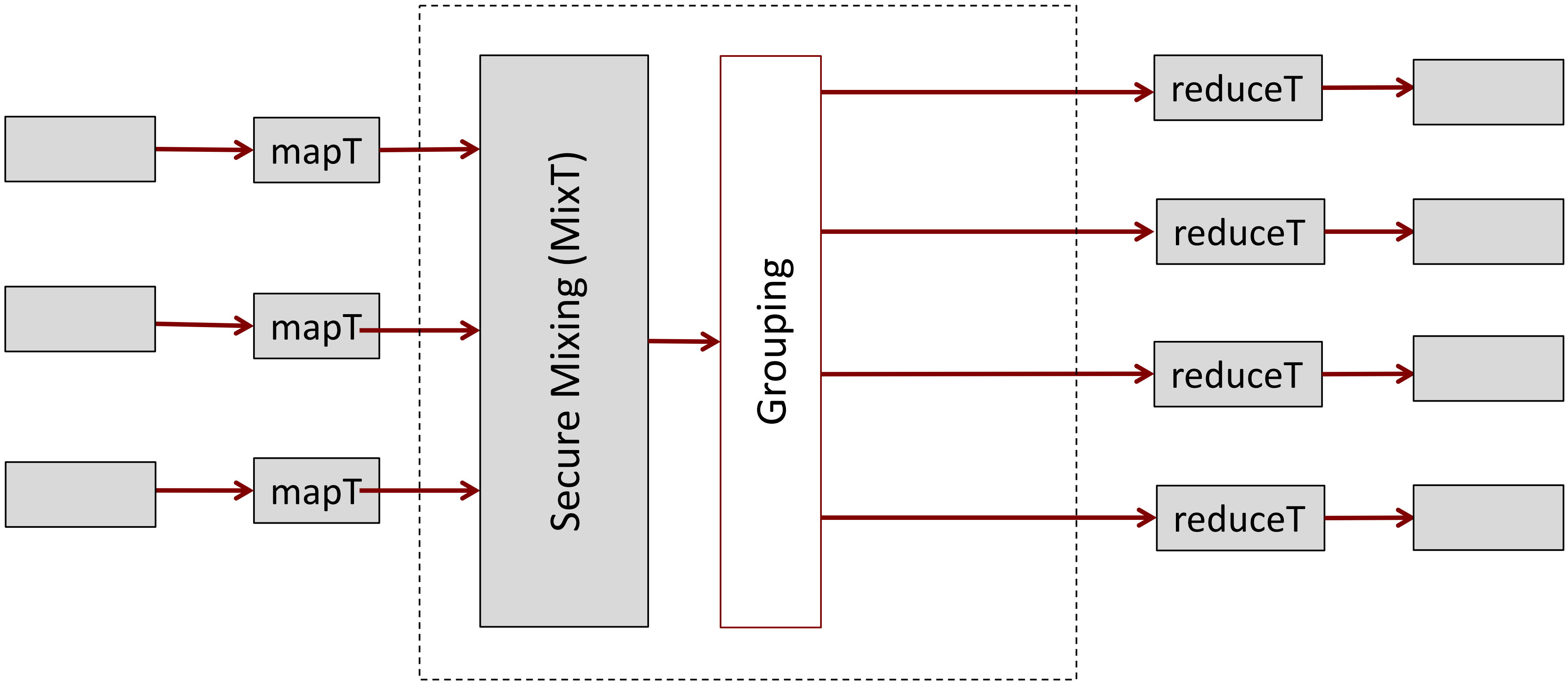
Possible solution: Oblivious RAM

- Very high overhead.



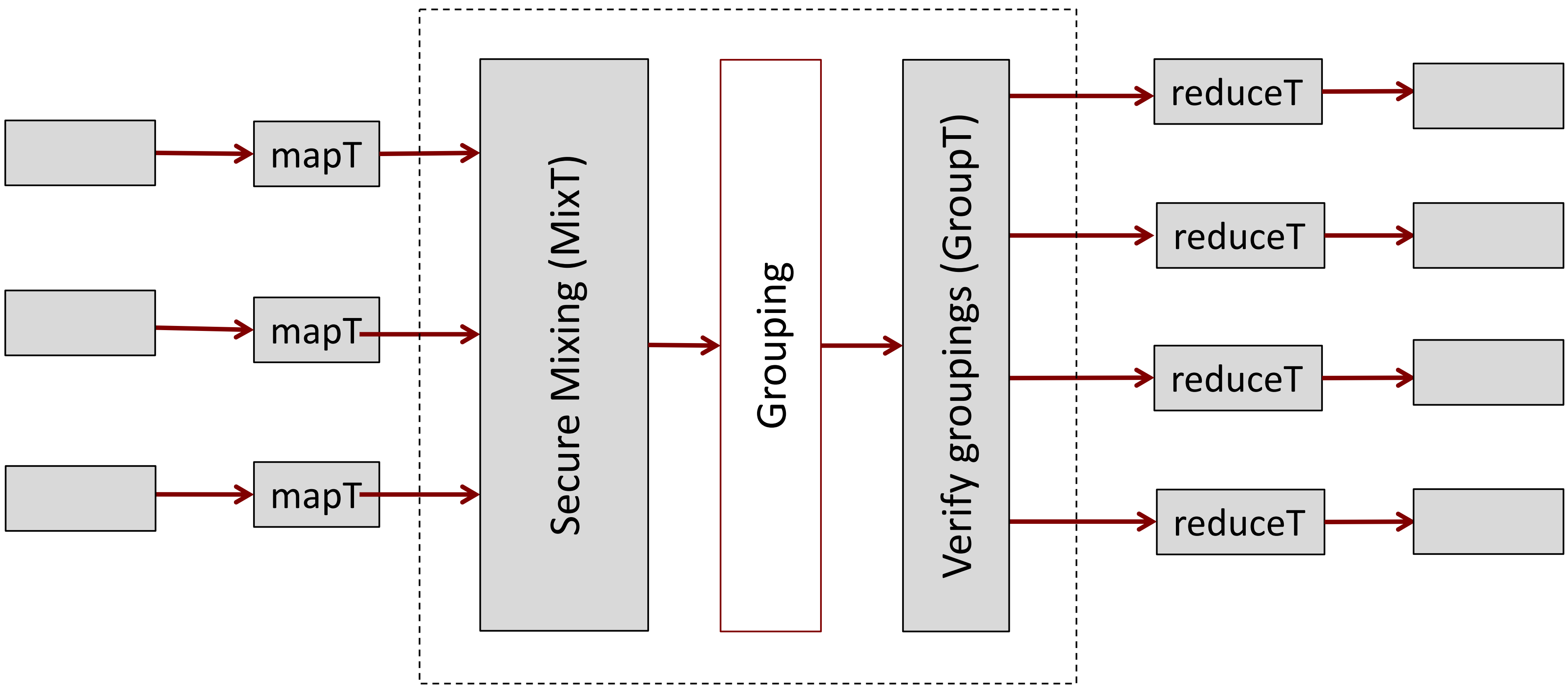
2. Our solution

- Randomly permutes the tuples.
- Group the tuples according to their keys.

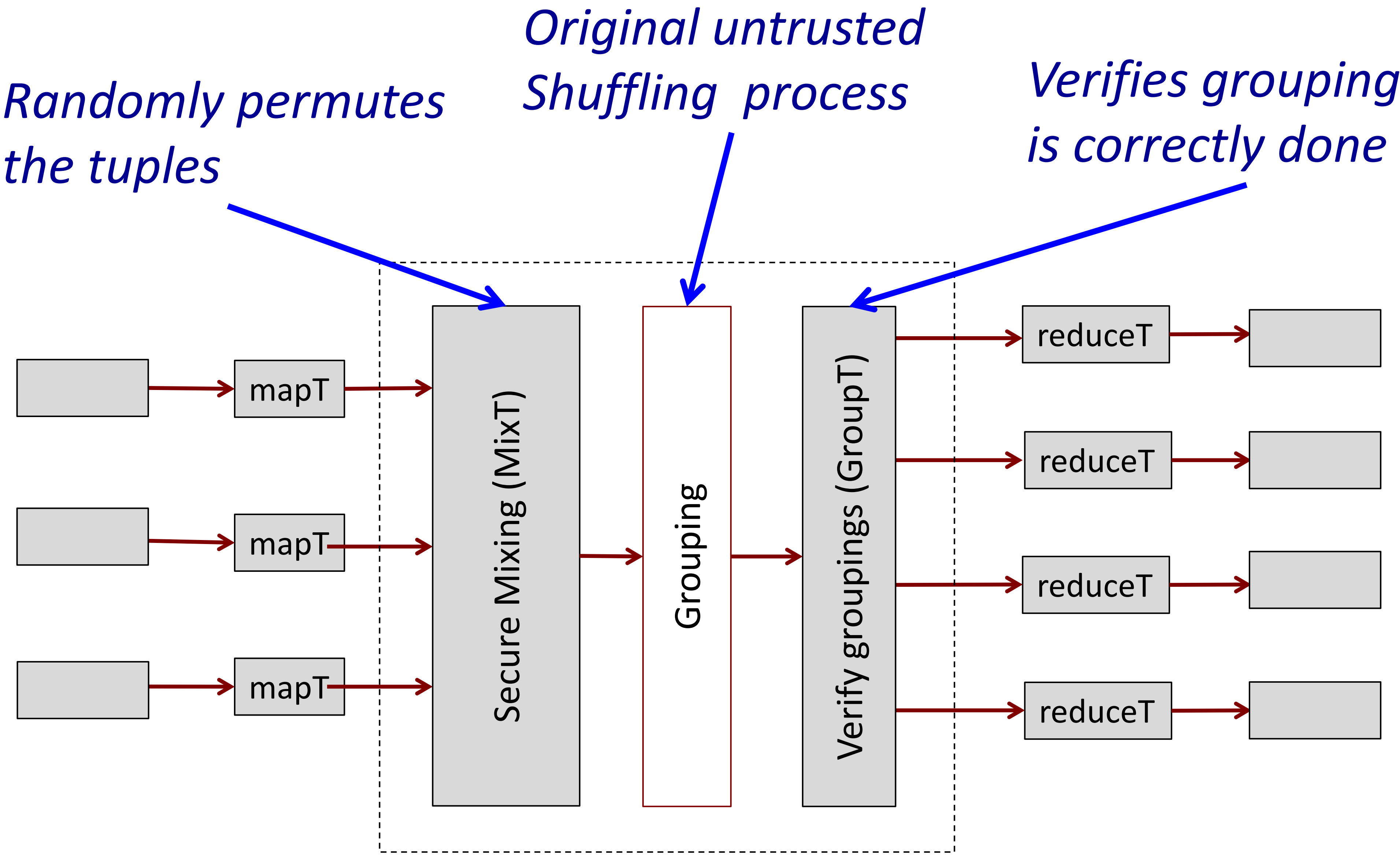


2. Our solution

- For execution integrity, addition step of verification is required.

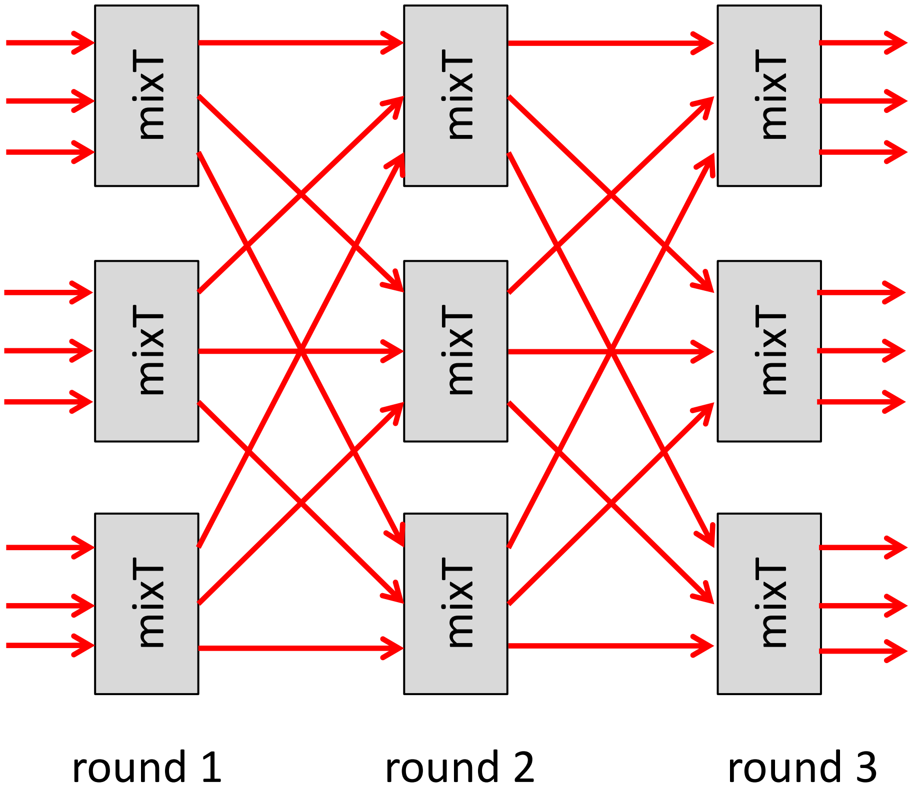


2. Our solution



Cascaded Mixing

A *cascaded mixing* is employed to randomly permute the tuples distributedly.



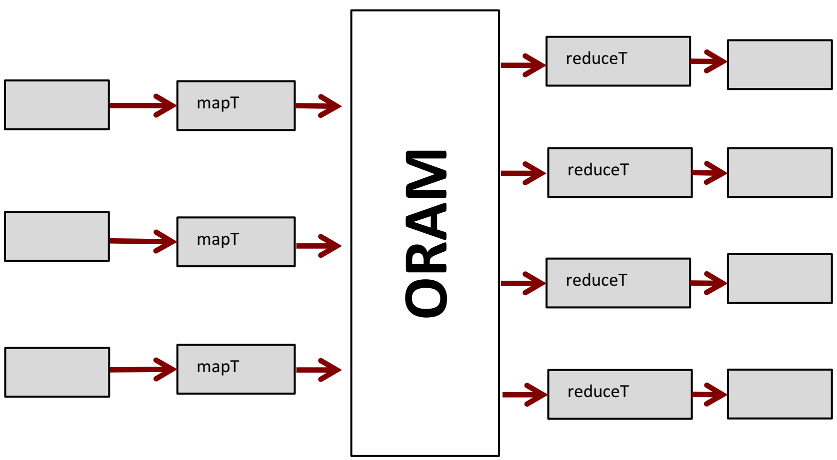
Remarks

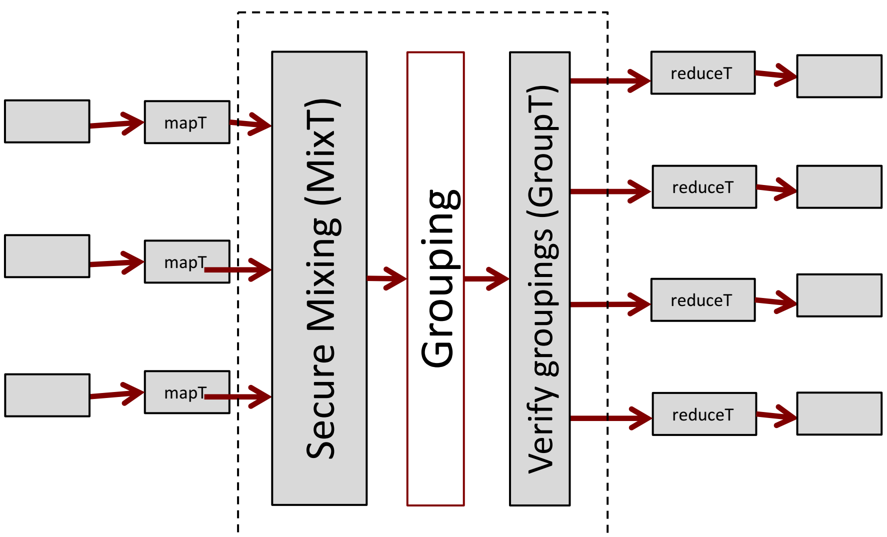
- Key management, handling of the random nonce and initial value is not straightforward.

- In Hadoop, multiple reduce instances are carried out by a single ***reducer***. Likewise ***mapper***.

ORAM vs Our solution

- M²R exploits the fact that, reads and writes can be "batched" into 2 phases, whereas ORAM caters for single read/write and thus incurs higher overhead.
- Many constructions of ORAM need to permute or o-sort the data.





3. Security Model

Adversary can observe the following:

- Input/output size of each trusted instance.
- Source/destination of the input/output.
- Time of invocation/return of each trusted instance.

Active adversary can:

- Arbitrary Invoke trusted instances.
- Halt instances.
- Drop/duplicate ciphertext (encrypted tuples).
- Add delays.

Modulo-Ψ private

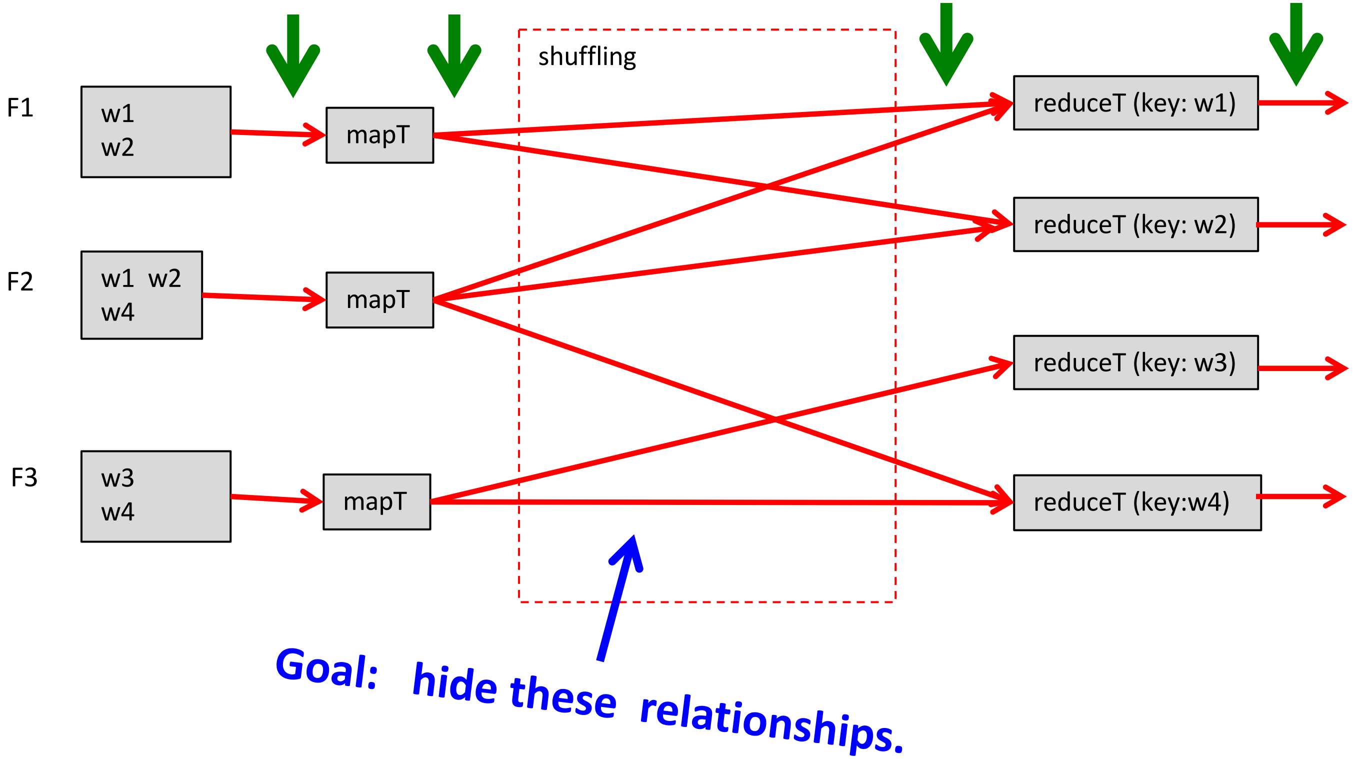
Based on formulation by Canetti (FOCS 01).

Let Ψ be the permissible data that can be revealed during honest execution.

A provisioning protocol is modulo-Ψ private if, for any adversary $\mathcal{A}$ executing the protocol, there is an algorithm $\mathcal{B}$ with access only to Ψ, such that the output of $\mathcal{A}$ and $\mathcal{B}$ are indistinguishable.

The permissible Ψ :

- size of input/output, time of revocation/return of mapT and reduceT under honest execution.



M^2R is Ψ-modulo private.

4. Implementations & Experiments

- Use Xen-4.3.3 as the trusted hypervisor, and its Verifiable Dynamic Function Executor to load and execute trusted codes. (The design of M^2R can be implemented differently depending on the underlying architecture, e.g. on Intel SGX).

- Ported 7 MapReduc benchmark applications.
  - KMeans : Iterative, Compute intensive
  - Grep : Compute intensive
  - Pagerank : Iterative, Compute intensive
  - WordCount : Shuffle intensive
  - Index : Shuffle intensive
  - Join : database queries
  - Aggregate : database queries

- 8 compute nodes, each quad-core Intel CPU 1.8 GHz, 8GB RAM, 1GB Ethernet cards.

Trusted Code Base

4 trusted computation units:

mixT, GroupT, mapT, reduceT.

- Platform related: (mixT, GroupT)

  Lines Of Code: ≈ 300
- Applications: (mapT, ReduceT)

  Lines Of Code ≈ 200 for our examples.

Performance

Job	Input size (bytes) (vs plaintext size)	Shuffled bytes	#Applications hyper calls	#platform hyper calls
Wordcount	2.1G (1.1×)	4.2G	3.3×10^6	35
Index	2.5G (1.2×)	8G	3.3×10^6	59
Grep	2.1G (1.1×)	75M	3.3×10^6	10
Aggregate	2.0G (1.2×)	289M	18.0×10^6	12
Join	2.0G (1.2×)	450M	11.0×10^6	14
Pagerank	2.5G (4.0×)	2.6G	1.7×10^6	21
KMeans	1.0G (1.1×)	11K	12.0×10^6	8

Running time (s)

Job	Baseline (vs no encryption)	M^2R (vs baseline)
Wordcount	570 (2.6 ×)	1156 (2.0 ×)
Index	666 (1.6 ×)	1549 (2.3 ×)
Grep	70 (1.5 ×)	106 (1.5 ×)
Aggregate	125 (1.6 ×)	205 (1.6 ×)
Join	422 (2 ×)	510 (1.2 ×)
Pagerank	521 (1.6 ×)	755 (1.4 ×)
KMeans	123 (1.7 ×)	145 (1.2 ×)

Conclusions

- Privacy-preserving distributed computation of MapReduce with trusted computing.
- Security:
  - Execution integrity +Data Confidentiality
  - Observation that simply running the map/reduce in trusted environment is not sufficient: interactions leak sensitive info.
  - Small TCB
- Exploit the algorithmic structure to outperform a solution that employs generic ORAM.
- Future works: other distributed dataflow systems.