CS 253 Cyber Security
Microarchitectural Security

ShanghaiTech University

The processor

Part of the trusted computing base (TCB):

- but is optimized for performance,
  … security may be secondary

Processor design and security:

- Important security features: hardware enclaves,
  memory encryption (TME), RDRAND, and others.
- Some features can be exploited for attacks:
  - Speculative execution, transactional memory, …

PART ONE

01

Trusted Execution Environment (TEE)

Intel SGX / Intel TDX

SGX / TDX: Goals

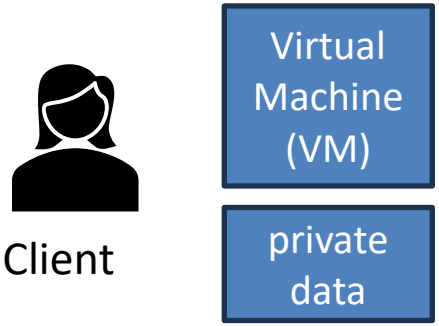
Extension to Intel processors that support:

- **Enclaves**: running code and memory isolated from the rest of system (code outside of enclave cannot read enclave memory)
- **Attestation**: prove to a remote system what code is running in the enclave
- **Minimum TCB**: only processor is trusted, nothing else!

  RAM and peripherals are untrusted

  ⇒ Memory controller must encrypt all writes to RAM (TME)

Why enclaves 1: cloud computing

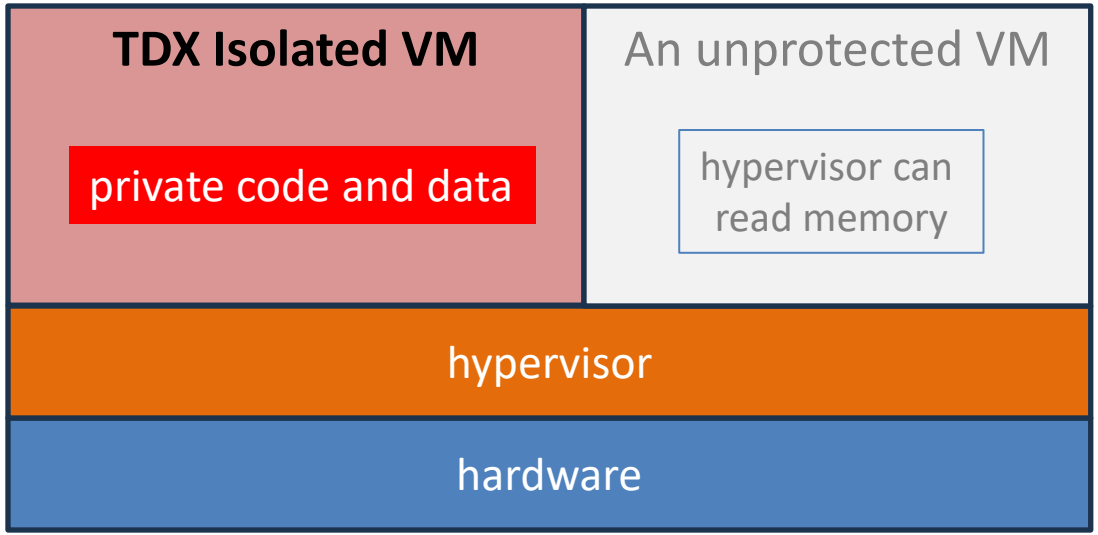


Goal: move data & VM to the cloud, without cloud seeing them in the clear

- Simple solution: encrypt data and VM, key stays with Client
- The problem: now cloud cannot search or compute on data

  ⇒ defeats the purpose of cloud computing

A solution: a HW enclave

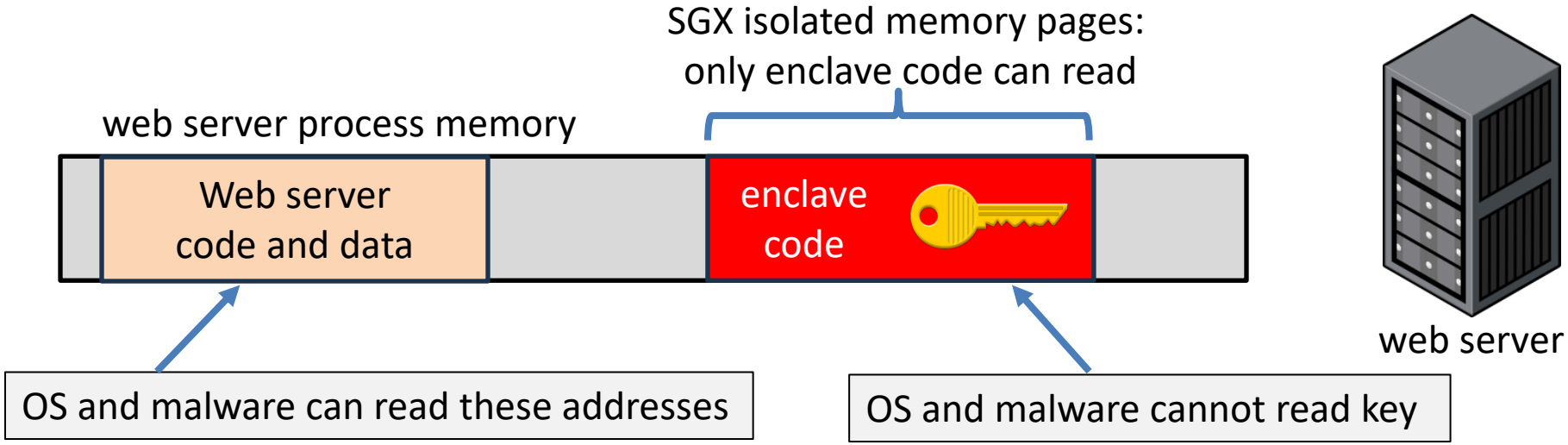


Cloud
(AWS, GCP, …)

Goal: hypervisor cannot read the memory of the isolated VM

How does this work? Will see in a minute.

Why enclaves 2: protecting keys



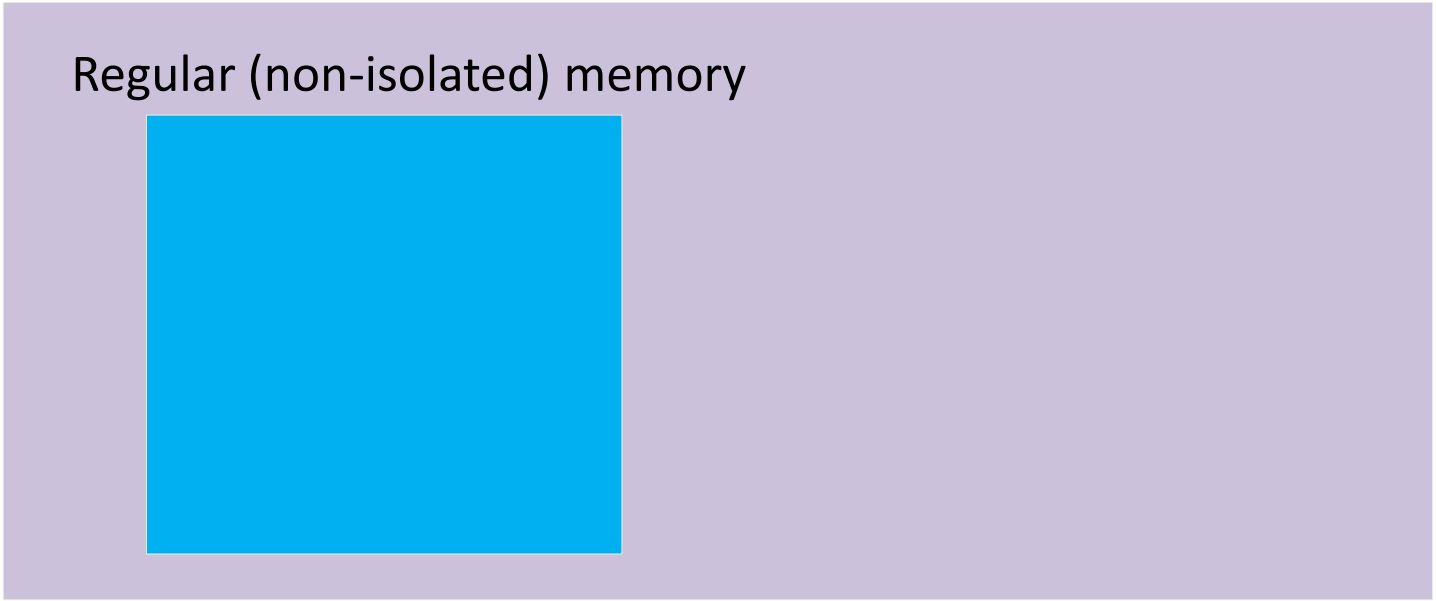
Storing a Web server HTTPS secret key:

secret key is only available in the clear inside an enclave

⇒ malware cannot extract the key

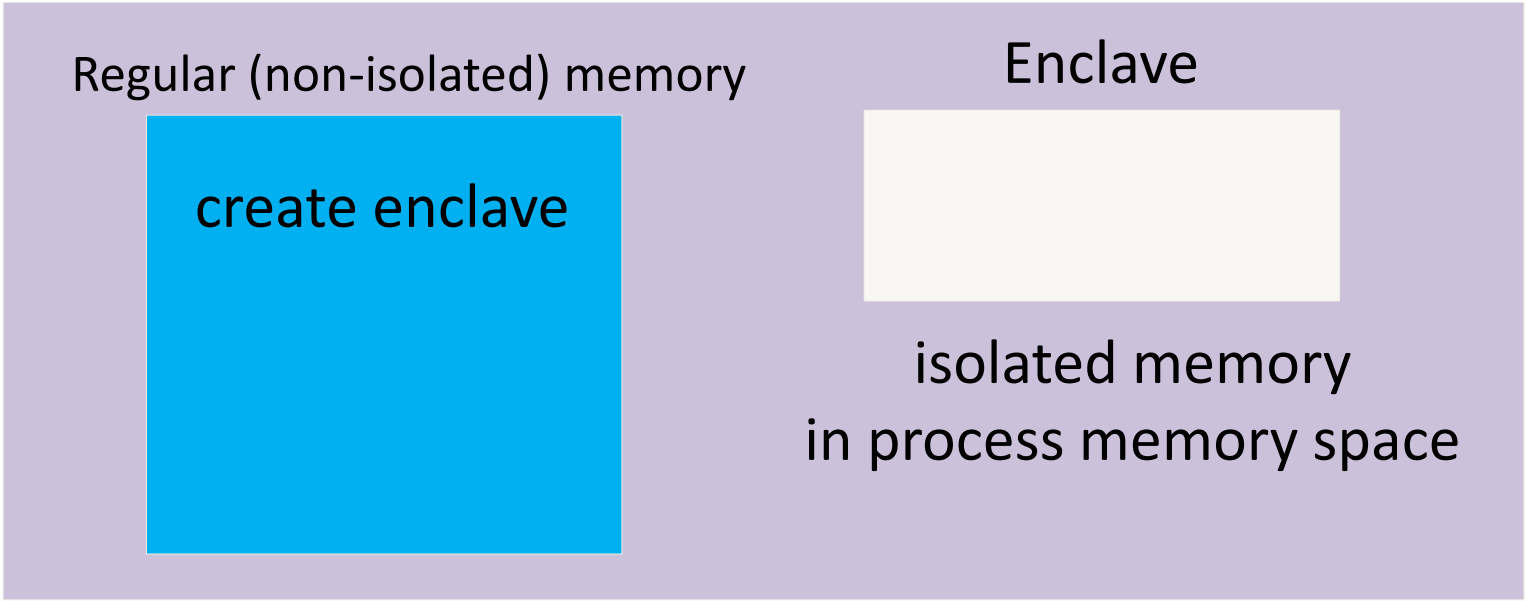
Intel SGX: how does it work?

An application defines part of itself as an enclave



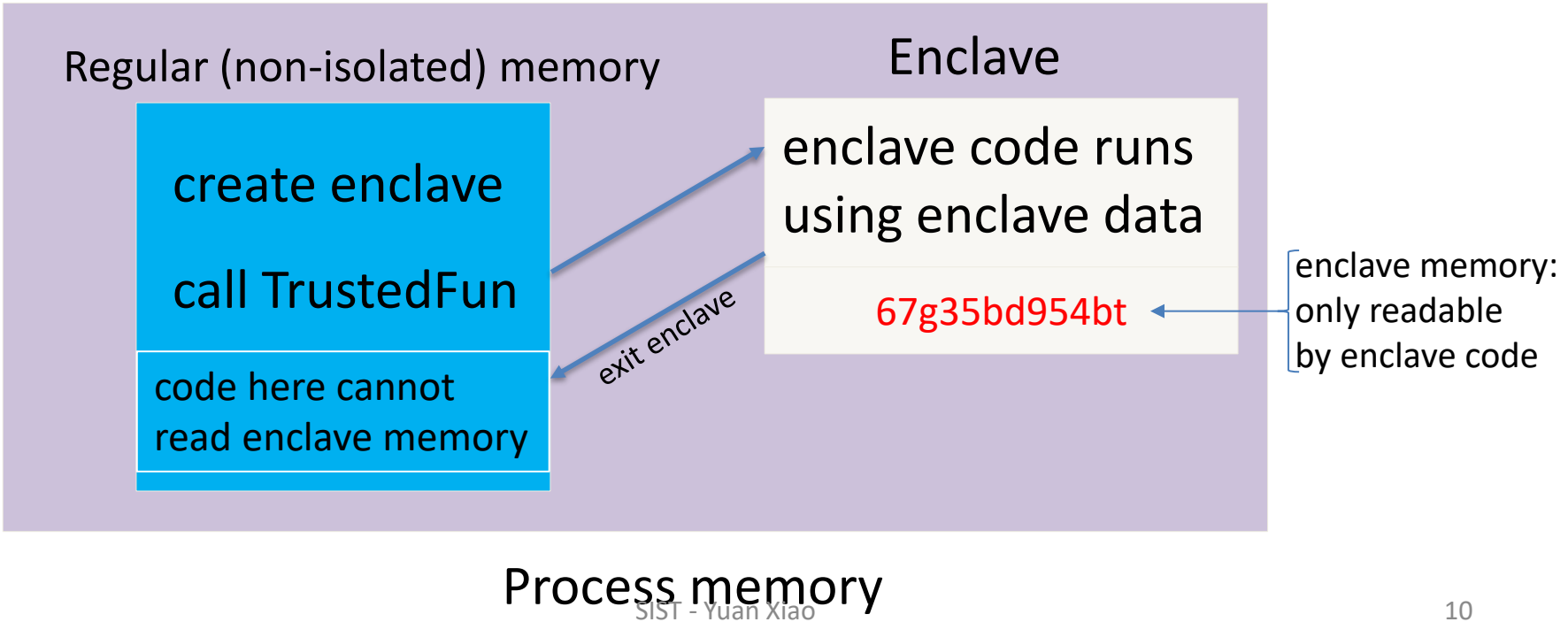
How does it work?

An application defines part of itself as an enclave



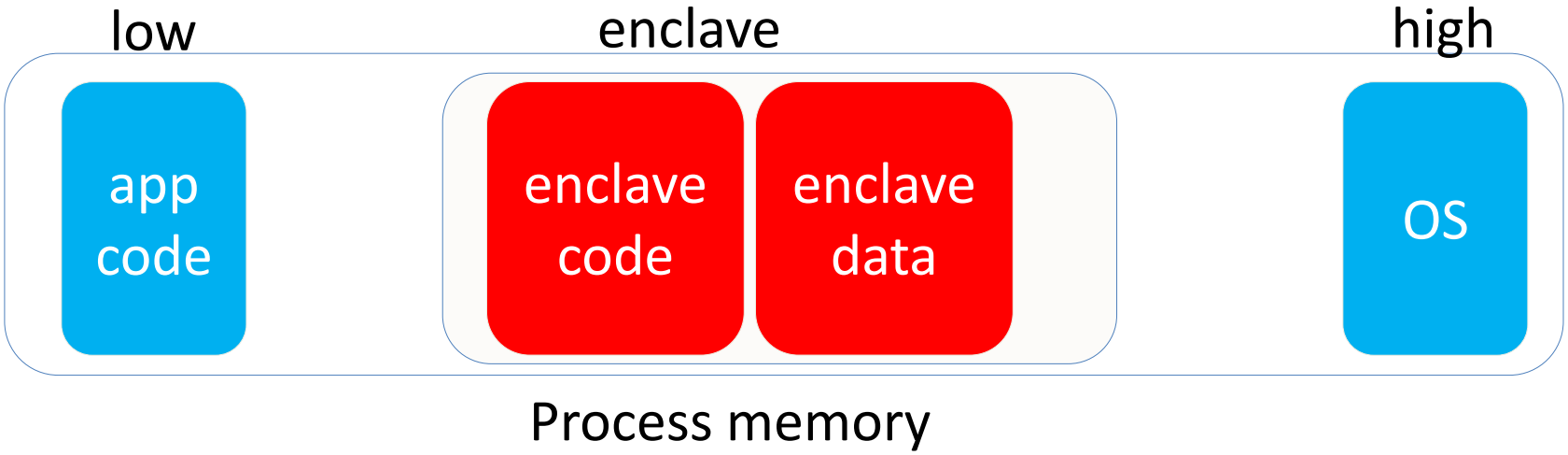
How does it work?

An application defines part of itself as an enclave



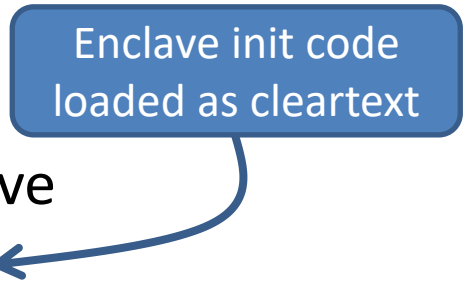
How does it work?

Part of process memory holds the enclave:



- Processor prevents access to cached enclave data outside of enclave.
- Enclave code and data are written **encrypted to RAM (TME)**

Creating an enclave: new instructions

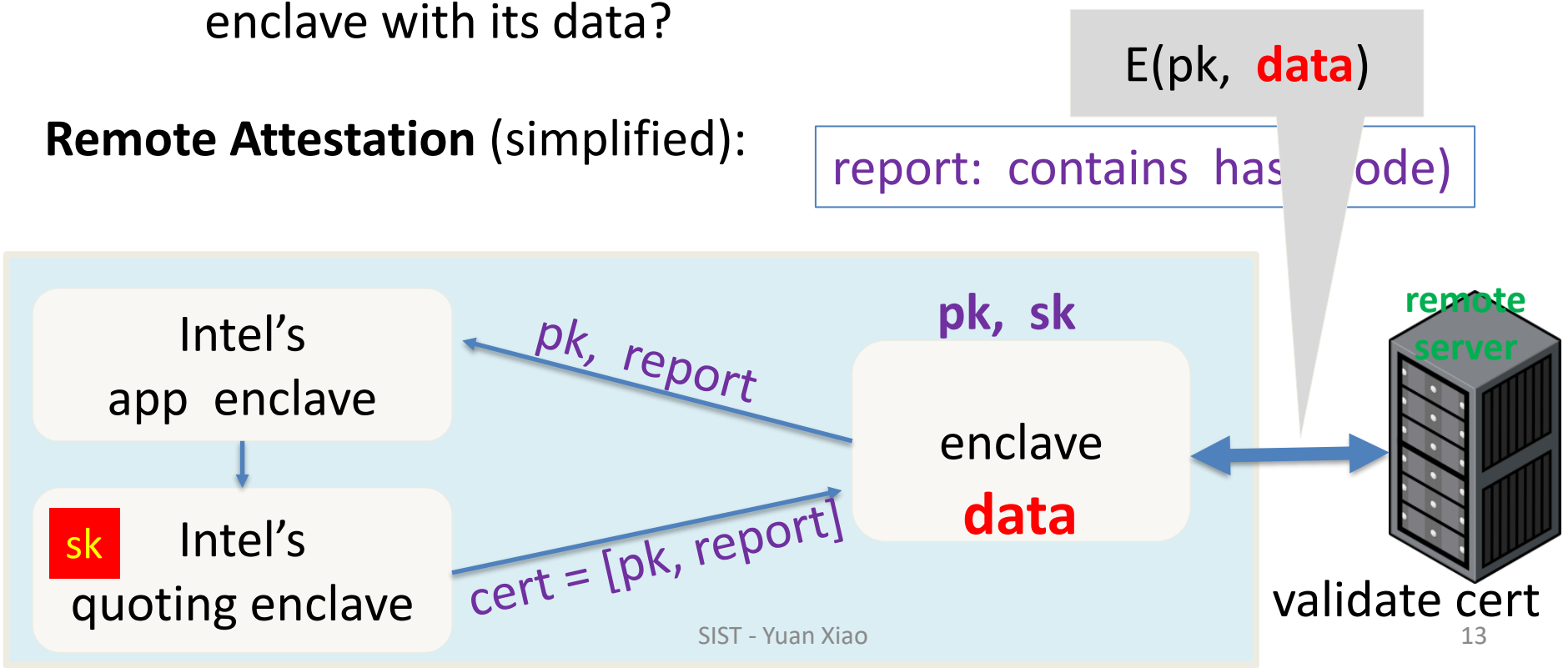


- **ECREATE**: establish memory address for enclave
- **EADD**: copies memory pages into enclave
- **EEXTEND**: computes hash of enclave contents (256 bytes at a time)
- **EINIT**: verifies that hashed content is properly signed
  if so, initializes enclave (signature = RSA-3072)
- **EENTER**: call a function inside enclave
- **EEXIT**: return from enclave

When to send secret data to enclave: attestation

The problem: How does a remote system know when it can trust an enclave with its data?

Remote Attestation (simplified):



SGX Summary

An architecture for managing secret data

- Intended to process data that cannot be read by anyone, except for code running in enclave
- Attestation: proves what code is running in enclave
- Minimal TCB: nothing trusted except for main processor
  ⇒ Memory controller encrypts all writes to RAM
- **Not suitable for legacy applications:** must split app into parts
  - Requires lots of code rewriting … not suitable for legacy apps

TDX Briefly: an easy-to-use encalve

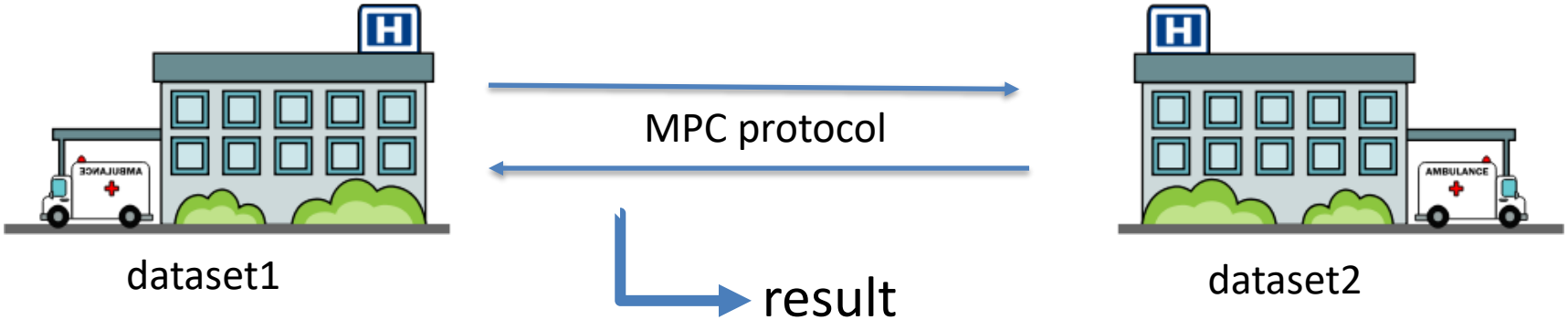
TDX: puts an entire VM in an enclave (e.g., an entire web server)

- Support for attestation and minimal TCB (as with SGX)
- Isolated VMs are managed by a new ***Intel TDX Module***
  - The TDX module is implemented in signed code by Intel
  - It is loaded into an isolated region of physical memory
  - Creates, manages, and attests to isolated VMs

https://cdrdv2.intel.com/v1/dl/getContent/690419

One more example application

Data science on federated data:

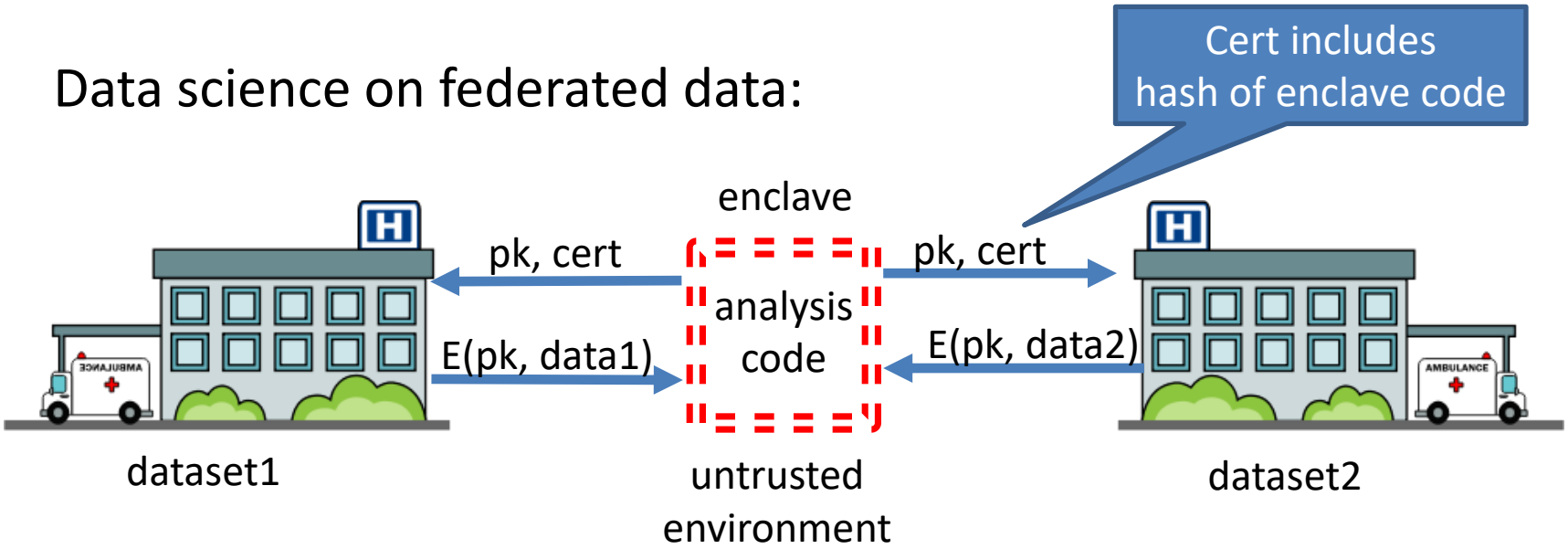


Can we run analysis on union(dataset1, dataset2) ??

cryptographic solutions (e.g., MPC) work for simple computations

An example application

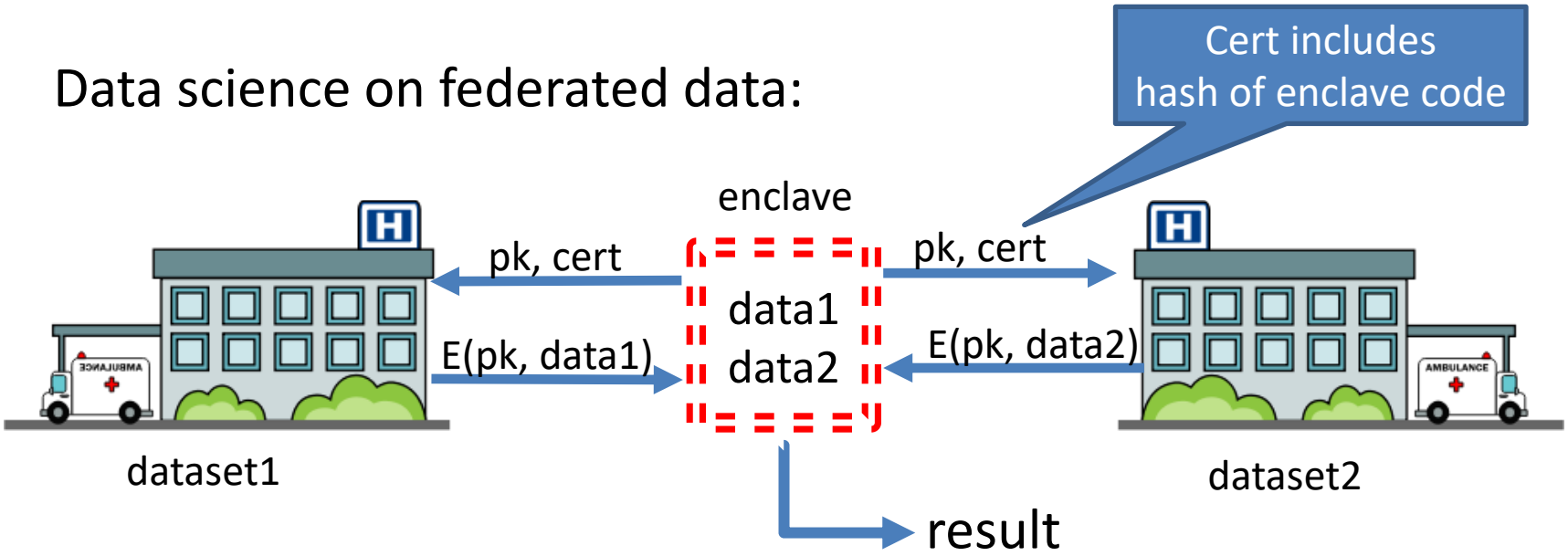
Data science on federated data:



For more complex analysis, can use (secure) hardware enclave

An example application

Data science on federated data:



For more complex analysis, can use (secure) hardware enclave

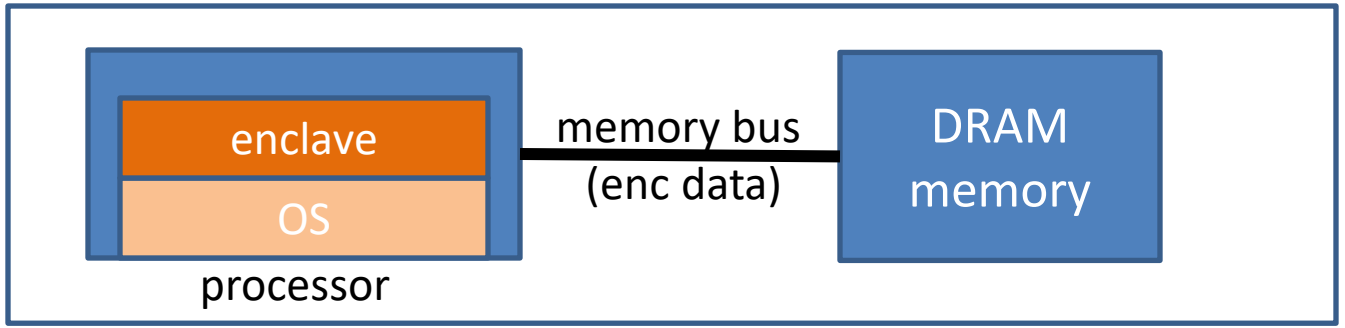
PART TWO

02

Side-channel Attacks

TDX insecurity: side channels

Also applies to SGX etc.



Attacker controls the OS. OS sees lots of side-channel info:

- Memory access patterns
- State of processor caches as enclave executes
- State of branch predictor

All can leak enclave data. Difficult to block.

Side Channel Attacks

- You are a dumb spy with no skills in lockpicking or window-prying.
- Meanwhile, you are a genius spy because you know how to infer the activities of your targets:
  - Light on and car in garage means at home
  - High kitchen temperature and ventilation noise means cooking
  - More than one shadows means having guests

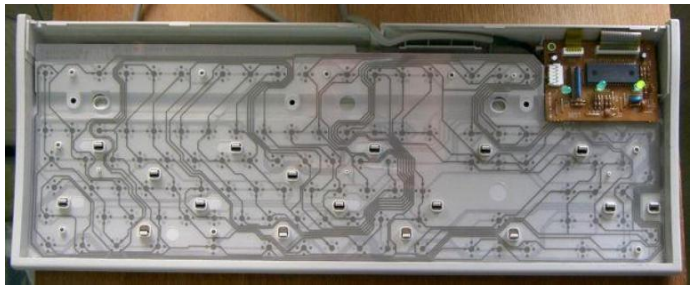
Extracting Information from Side Channels

- Learn from a case study:
  Inferring words typed on the keyboard by analyzing the sound

Keyboard Acoustic Emanations Revisited, Li Zhuang, Feng Zhou, J. D. Tygar, CCS 2005

Intuition: Why could this possibly work?

- Different keystrokes make different sounds
  - Locations
  - Underlying hardware



Threat Model and Challenges

- Attacker has a microphone recording the victim's typing
  - Assumptions: typing English text, no labeled input
  - Goals: recovering the English text, inferring random text (e.g., password)
- Challenges
  - Hard to obtain labeled training data --- no cooperation from the victim
  - Typing patterns can be keyboard specific
  - Typing patterns can be user specific

Threat Model and Challenges

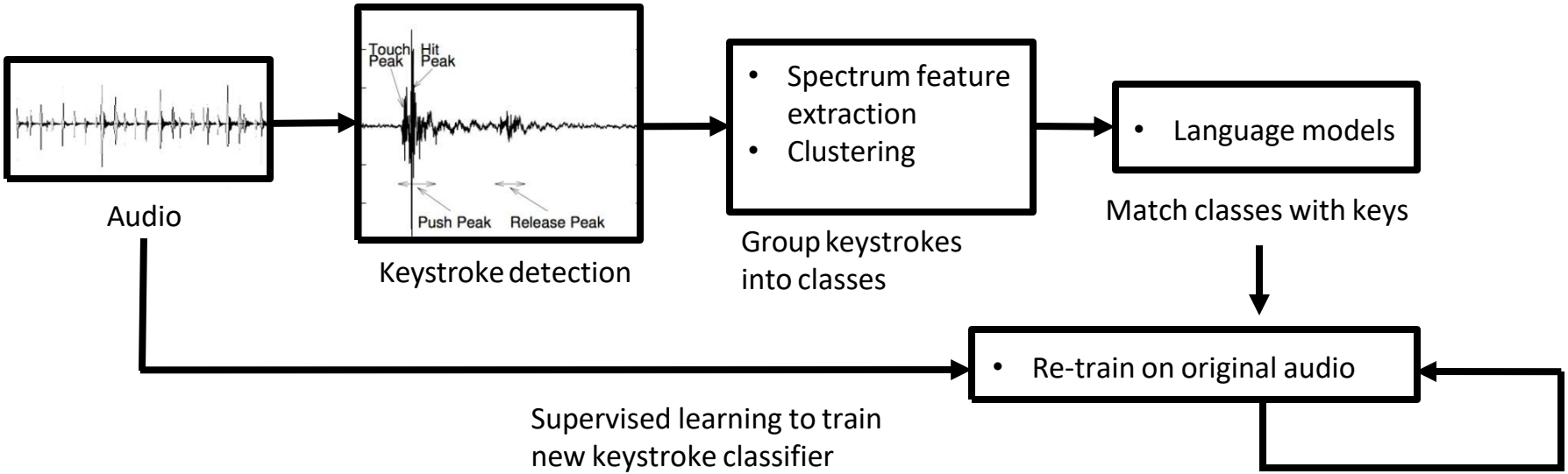
- Attacker has a microphone recording the victim's typing
  - Assumptions: typing English text, no labeled input
  - Goals: recovering the English text, inferring random text (e.g., password)
- Challenges

Key Intuition: the typed text is often not random.

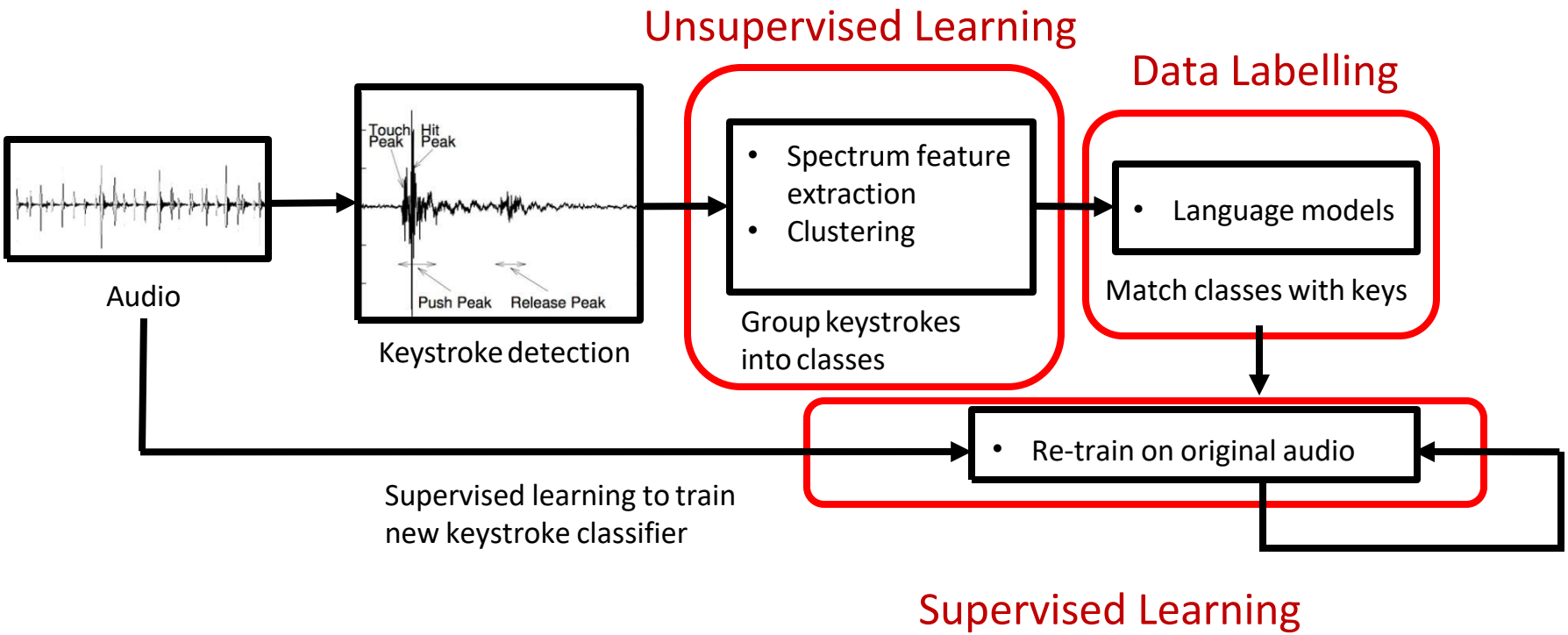
- English words limits the possible temporal combinations of keys
- English grammar limits the word combinations.

How The Attack Works

- Key idea: generating training data automatically
  - Labelling the audio of a key stroke with the actual key



A Combination of Different Learning Methods



Step1: Unsupervised Learning

- Unsupervised clustering
  - Feature generation
    - Cepstrum features
  - Clustering into K classes
    - K > N (actual number of keys used)
- Output
  - K unlabeled classes

- Spectrum feature extraction
- Clustering

Group keystrokes into classes

this is the best pizza in town

this is the best pizza in town

Step 2: Context-based Language Model

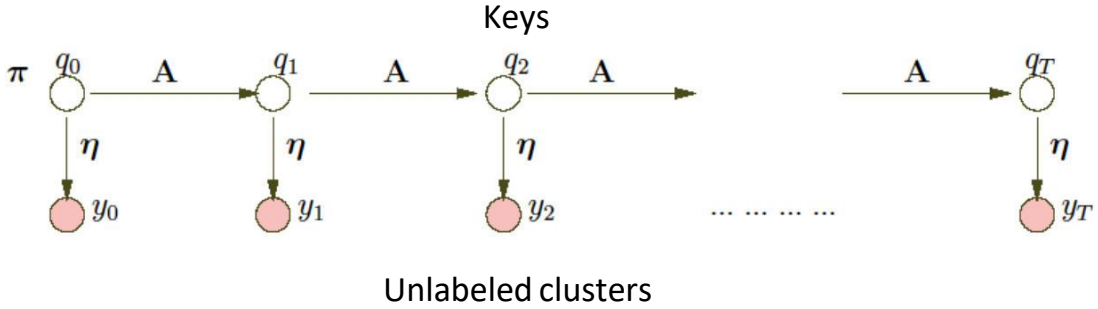
- Need to label the clusters: which key they represent?

- Assume the victim is typing English text
  - Characters follow certain frequency
  - Actual content follows English spelling and grammar

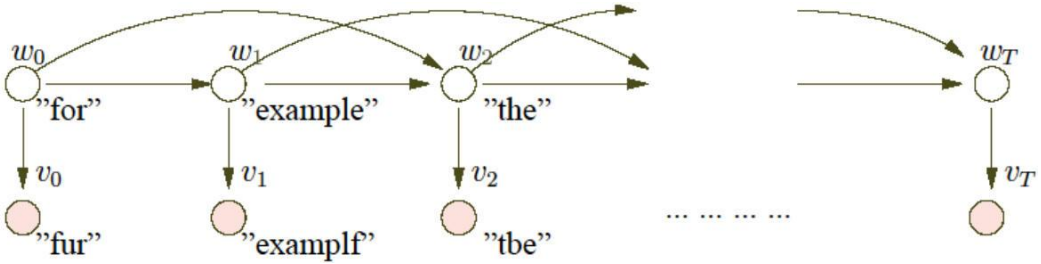
- Advantages:
  - Use 2-character combination frequency to match classes to keys
  - Use language model (spelling, grammar) to correct mistakes

Details: Context-based Language Model

- Character-level mapping:
  - Hidden Markov Model
  - Produce a probability of keys assigned to classes.
  - Example: “th” vs. “tj”



- Word-level correction:
  1. Spell check
  2. Grammar
     - Tri-gram



Details: Context-based Language Model

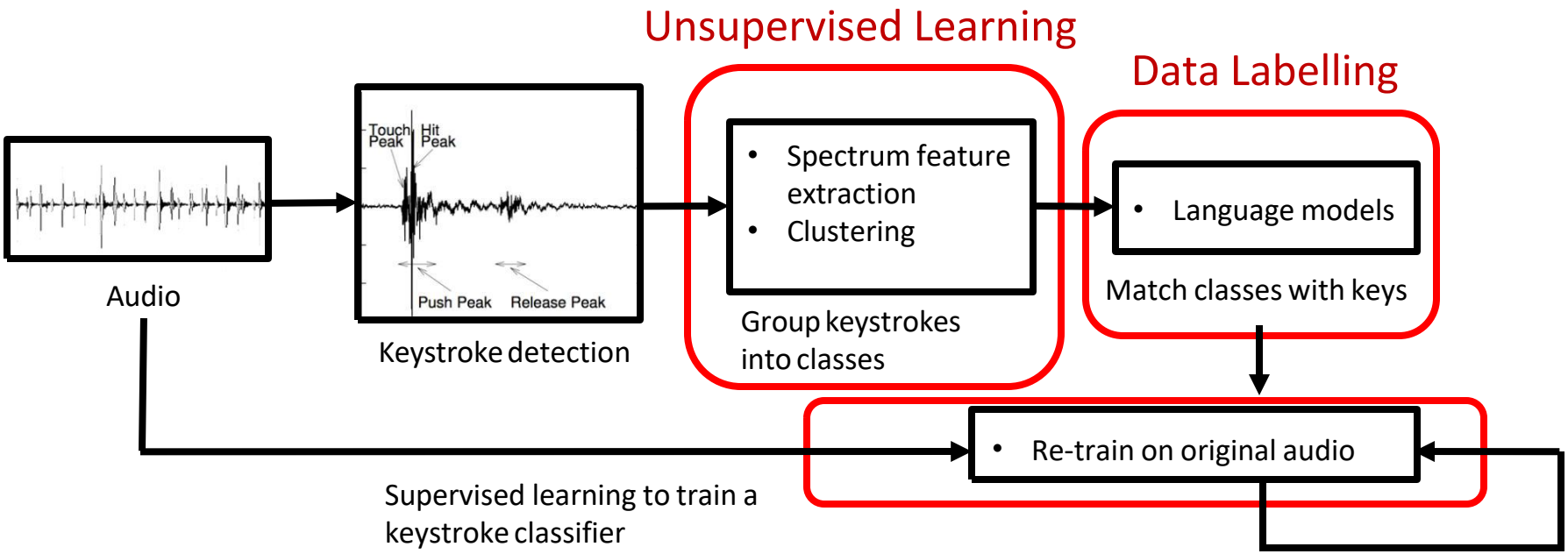
Before spelling and grammar correction

the big money fight has drawn the shoporo od dosens of companies in the entertainment industry as well as attorneys gnnerals on states, who fear the fild shading softwate will encourage illegal acylvitt, srem the grosth of small arrists and lead to lost cobs and dimished sales tas revenue.

After spelling and grammar correction

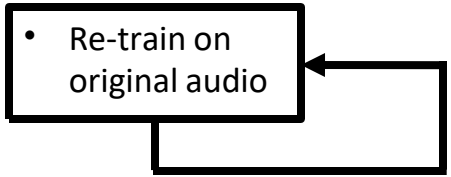
the big money fight has drawn the support of dozens of companies in the entertainment industry as well as attorneys generals in states, who fear the film sharing software will encourage illegal activity, stem the growth of small artists and lead to lost jobs and finished sales tax revenue.

A Combination of Different Learning Methods

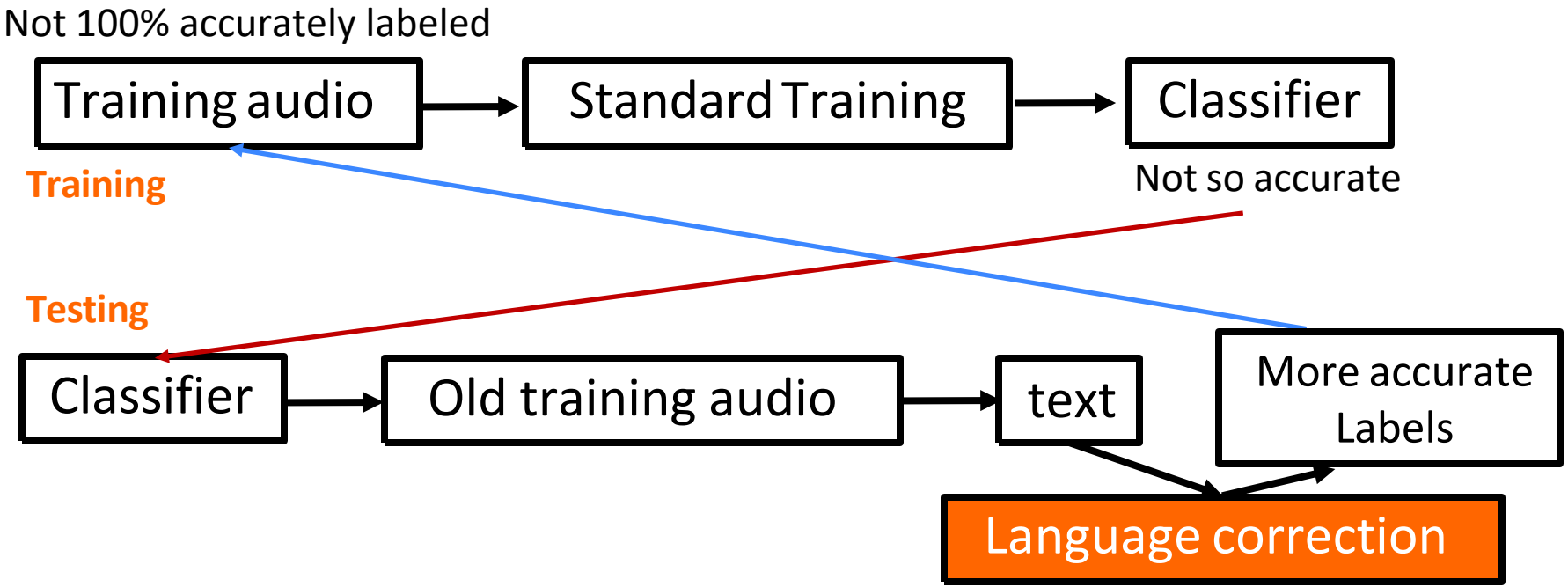


Feedback based Training

- A keystroke classifier (for inferring random text)
  - Given a keystroke, produce the label of the key
- Training
  - Input: noisy training data
    - Only a subset of labeled data from the language models
    - Choose those with fewer corrections by the language model (quality indicator)
  - Output: a not so accurate keystroke classifier
- Testing
  - Use the trained classifier to classify the training data again
  - Use the language model to correct the classification result
  - Use the corrected label for re-training



Feedback based Training (Con't)



Evaluation

		Set 1		*Set 2*		*Set 3*		*Set 4*	
		words	chars	words	chars	words	chars	words	chars
unsupervised learning	keystrokes	34.72	76.17	38.50	79.60	31.61	72.99	23.22	67.67
	language	74.57	87.19	71.30	87.05	56.57	80.37	51.23	75.07
1st supervised feedback	keystrokes	58.19	89.02	58.20	89.86	51.53	87.37	37.84	82.02
	language	89.73	95.94	88.10	95.64	78.75	92.55	73.22	88.60
2nd supervised feedback	keystrokes	65.28	91.81	62.80	91.07	61.75	90.76	45.36	85.98
	language	90.95	96.46	88.70	95.93	82.74	94.48	78.42	91.49
3rd supervised feedback	keystrokes	66.01	**92.04**	62.70	**91.20**	63.35	**91.21**	48.22	**86.58**
	language	**90.46**	**96.34**	**89.30**	**96.09**	**83.13**	**94.72**	**79.51**	**92.49**

Table 2: Text recovery rate at each step. All numbers are percentages.

Other Key Results

- Works for random text
  - Inferring passwords that contain English letters only
  - 90% of 5-character random passwords: < 20 attempts
  - 80% of 10-character random passwords: <75 attempts

- Works for multiple types of keyboards
- Even “low-quality” microphones can do the job

Possible Defenses

- Introduce noise into the system
  - Add (random) background noise to keystrokes
    - Remove the unique pattern for each key
  - Use quieter keyboards

- Other defenses
  - Two factor authentication (not just typing a password)
  - No microphone in your room?

THE END