# Information Security: Theory vs. Reality

## 0368-4474, Winter 2015–2016

## Lecture 4:

# Machine Learning Techniques in Side-Channel Analysis

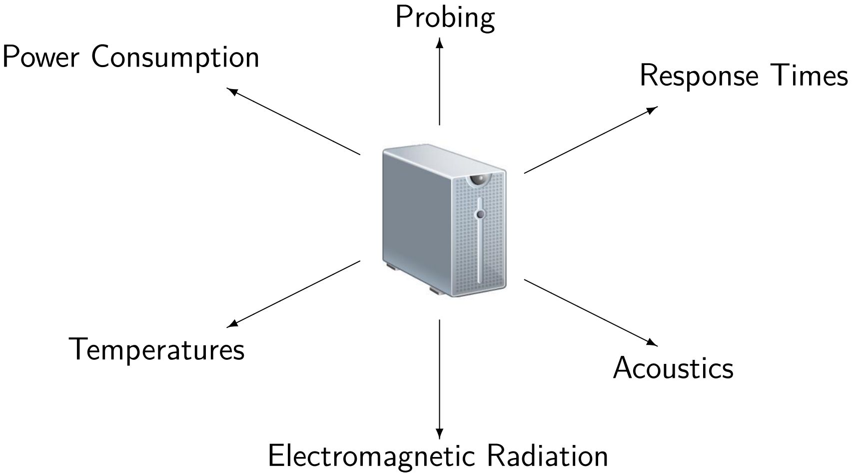
Lecturer: Eran Tromer

Including presentation material by Guy Wolf

# Outline

# Side-Channel Leaks

# Side-Channel Leaks



# Processing Leaked Data

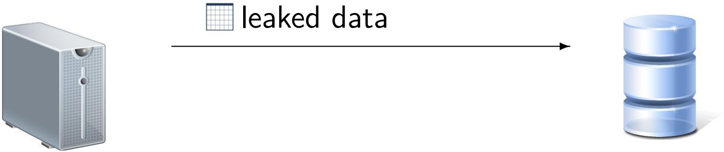
**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.

# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.

# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.

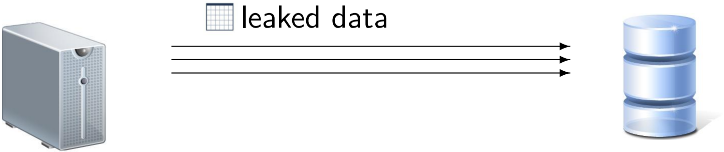


# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.

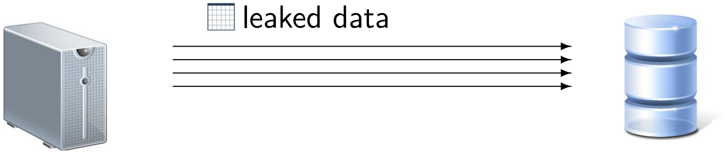
# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.



# Processing Leaked Data

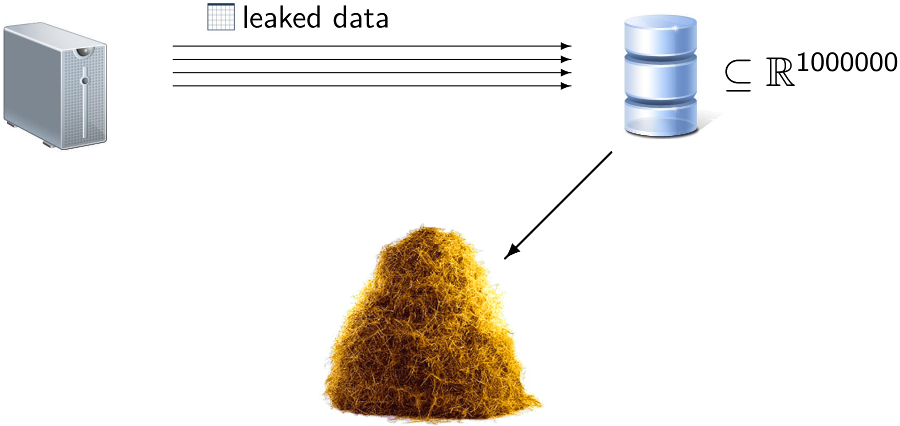
**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.




$\subseteq \mathbb{R}^{1000000}$

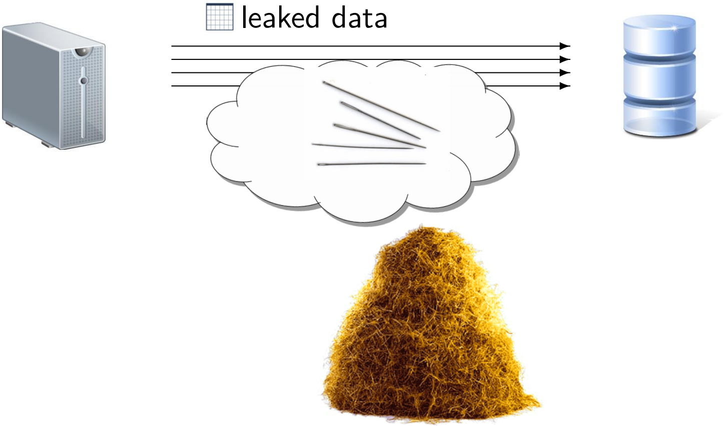
# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.



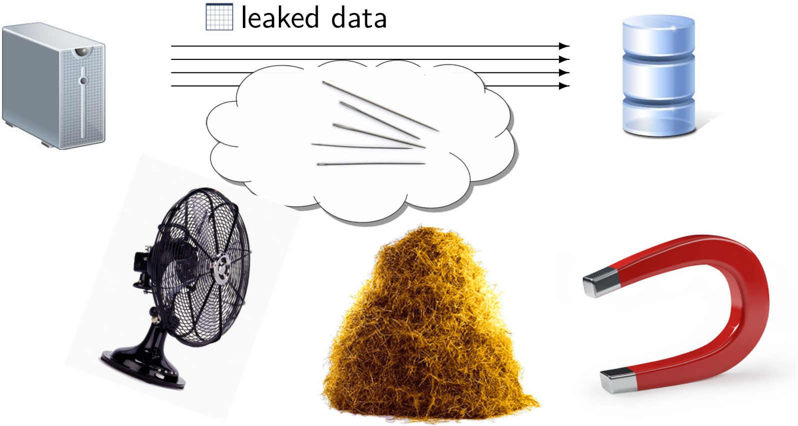
# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.



# Processing Leaked Data

**Traces** are vectors representing a measured physical quantity as a function of time, during the attacked operation. They containing hundreds or more (often millions) of measurement points.



# Machine Learning

Machine learning encompasses tools that perform smart analysis of data, such as:

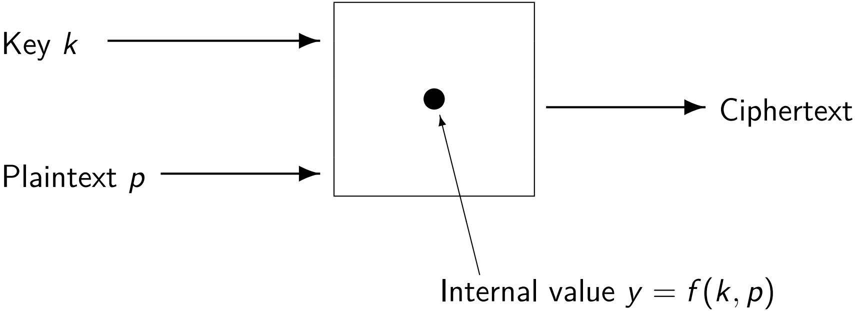
- Discovery of useful, possibly unexpected, patterns in data
- Non-trivial extraction of implicit, previously unknown and potentially useful information from data
- Exploration & analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns

Common tasks include:

- Dimensionality reduction
- Clustering & Classification
- Regression & Out-of-sample extension

# Quick Recap:
# Correlation Power Analysis

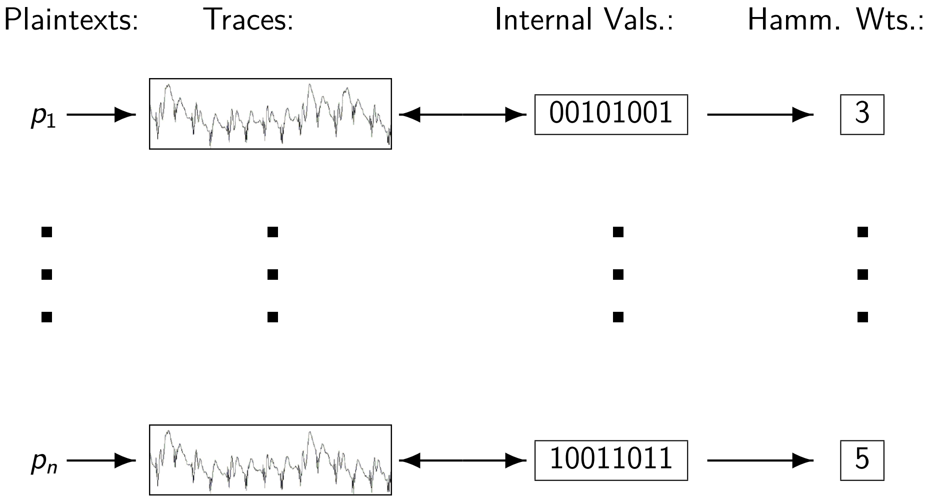
# Quick Recap: Correlation Power Analysis




**Typical assumption:**

power consumption correlates with Hamming Weight of $y$

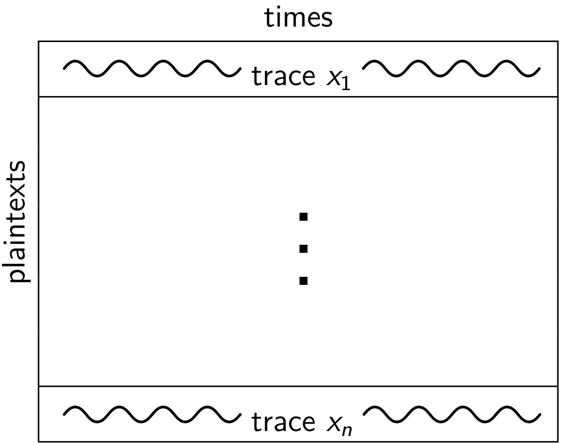
# Quick Recap: Correlation Power Analysis

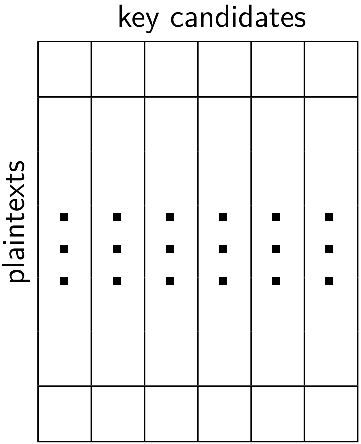


# Quick Recap: Correlation Power Analysis

Traces matrix:

Hamming Weight matrix:

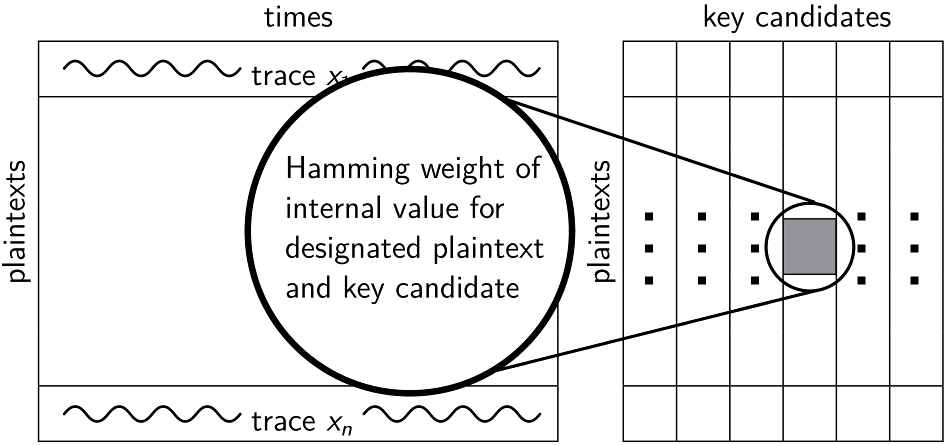


# Quick Recap: Correlation Power Analysis

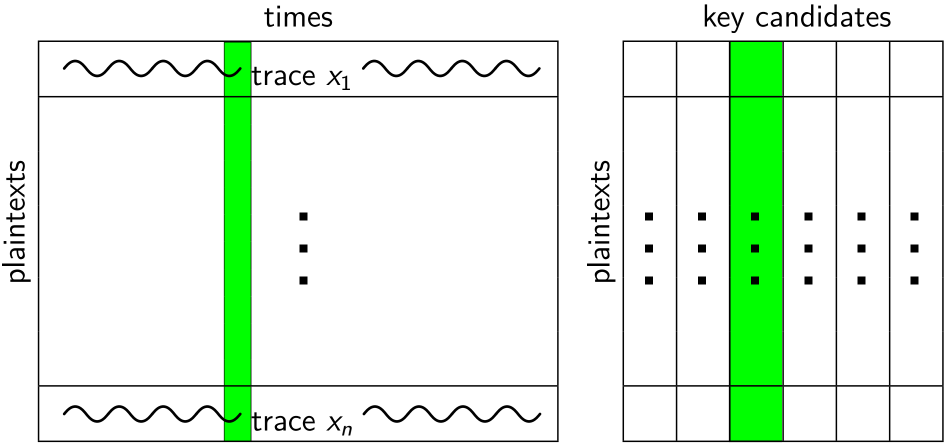
Traces matrix:

Hamming Weight matrix:



# Quick Recap: Correlation Power Analysis

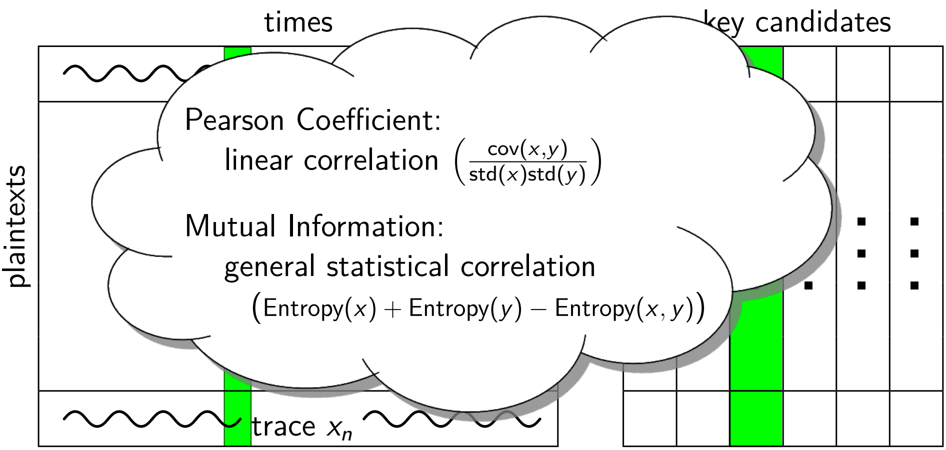
Traces matrix: Hamming Weight matrix:




Correlation indicates the correct time and key

# Quick Recap: Correlation Power Analysis

Traces matrix: Hamming Weight matrix:




Pearson Coefficient:

linear correlation $\left(\frac{\mathrm{cov}(x,y)}{\mathrm{std}(x)\mathrm{std}(y)}\right)$

Mutual Information:

general statistical correlation

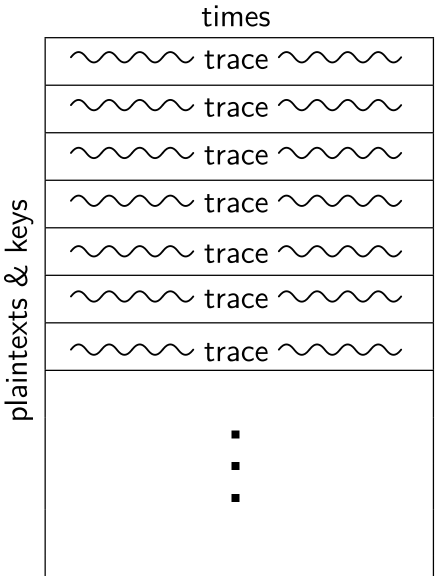
$\left(\mathrm{Entropy}(x) + \mathrm{Entropy}(y) - \mathrm{Entropy}(x, y)\right)$

Correlation indicates the correct time and key

# Alternative Attack: Template Power Analysis

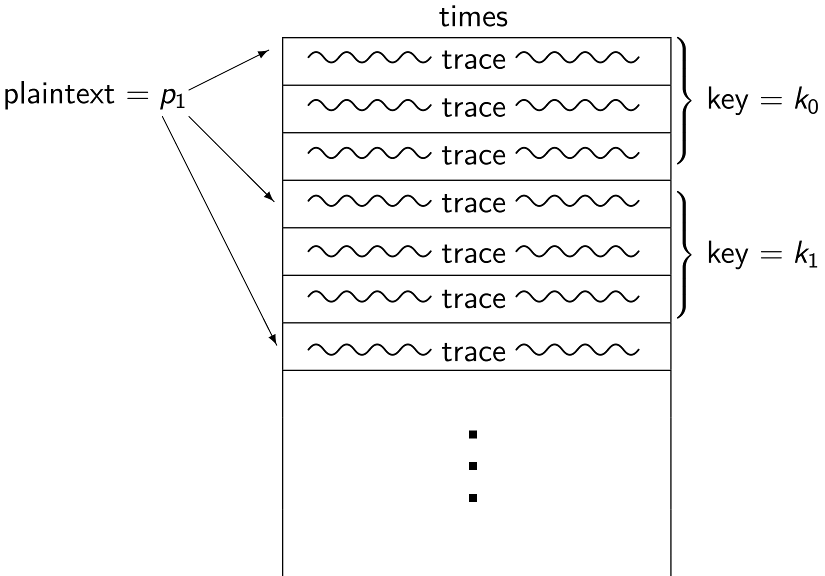
# Template Power Analysis

**Training**: Learn traces of many plaintexts & keys



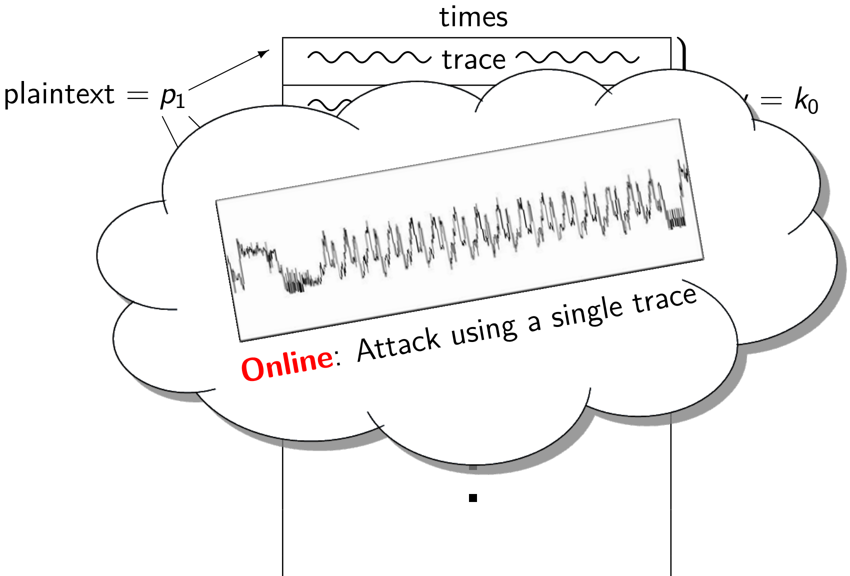
# Template Power Analysis

**Training**: Learn traces of many plaintexts & keys



# Template Power Analysis

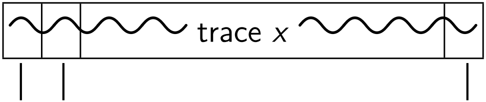
**Training**: Learn traces of many plaintexts & keys



# Template Power Analysis

## Representing power traces as vectors

Traces are high-dimensional vectors, containing hundreds or more (often millions) of measurement points.




$$\left( x_{[1]}, x_{[2]}, \quad \cdots \quad , x_{[m]} \right) \in \mathbb{R}^m$$

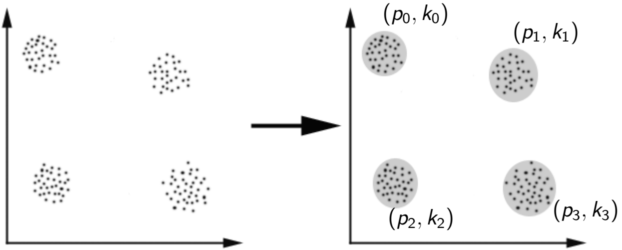
Traces can be analyzed as vectors in the Euclidean space $\mathbb{R}^m$ with $\ell_2$ norms & distances

- Somewhat arbitrary representation, but effective and convenient

# Template Power Analysis

## Learning & analyzing trace vectors

Template assumption: the positions of traces in $\mathbb{R}^m$ (with $\ell_2$ norm) correlate with the plaintext & key values that generated them




- Correlation is either directly seen or via some internal value
- Ideally, correlation is expressed as clustering by plaintext & key

Theoretically, can detect clusters using *machine learning* tools too. Usually, enough is known to classify traces into clusters; the challenge is to characterize these clusters geometrically so we can check which cluster the online trace resides

# Template Power Analysis

## Example: classic TPA attack using Gaussian statistical modeling[1]

Model power consumption of encrypting plaintext $p$ with key $k$ as a random variable $\vec{X}_{(k,p)} \sim \mathcal{N}(\vec{\mu}_{(k,p)}, \Sigma_{(k,p)})$

- $\vec{X}_{(k,p)}$ is drawn from a multidimensional normal (Gaussian) distribution
- $\vec{\mu}_{(k,p)} \in \mathbb{R}^m$ is the mean power consumption for $k$ & $p$
- $\Sigma_{(k,p)}$ is the $m \times m$ noise covariance matrix for $k$ & $p$
- The likelihood of a trace $\vec{x}$ originating from $k$ & $p$ is $\mathcal{L}_{(k,p)}(\vec{x}) = \left((2\pi)^m |\Sigma_{(k,p)}|\right)^{-1/2} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_{(k,p)})^T \Sigma_{(k,p)}^{-1} (\vec{x} - \vec{\mu}_{(k,p)})\right)$

Attack single trace (with known plaintext $p$):

- Compute likelihood of it originating from each key candidate (with $p$)
- Choose key candidate with maximum likelihood

[1] "Template Attacks", 2003, by S. Chari, J.R. Rao, and P. Rohatgi

# Template Power Analysis

## Curse of dimensionality

Directly analyzing high-dimensional vectors is usually **infeasible** due to:

### Curse of Dimensionality

A general term for various phenomena that arise when analyzing/organizing high-dimensional data.

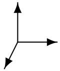
- Common theme - difficult/impractical/impossible to obtain statistical significance due to sparsity of the data in high-dimensions
- Causes poor performance (computational complexity)
- Causes poor results (bad estimates)

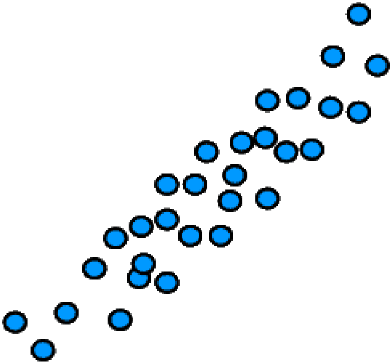
Common solution - use **dimensionality reduction** methods and analyze their resulting embedded space.

Example: only use the $\ell < m$ time indices that provide the highest differences between mean power consumptions of different key-plaintext pairs. This is traditional *differential power analysis*.

# Dimensionality Reduction with Principal Component Analysis

3D space

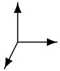


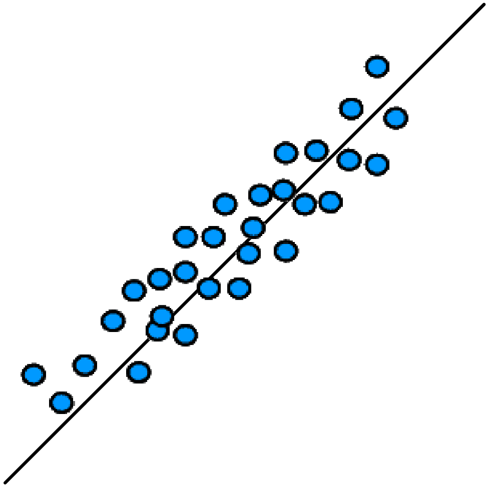


# Dimensionality Reduction

## Principal Component Analysis (PCA)

3D space



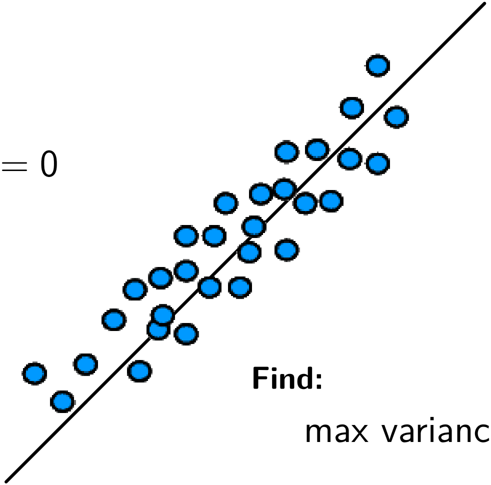


# Dimensionality Reduction

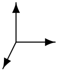
## Principal Component Analysis (PCA)

**Assume:**

$avg = 0$



3D space

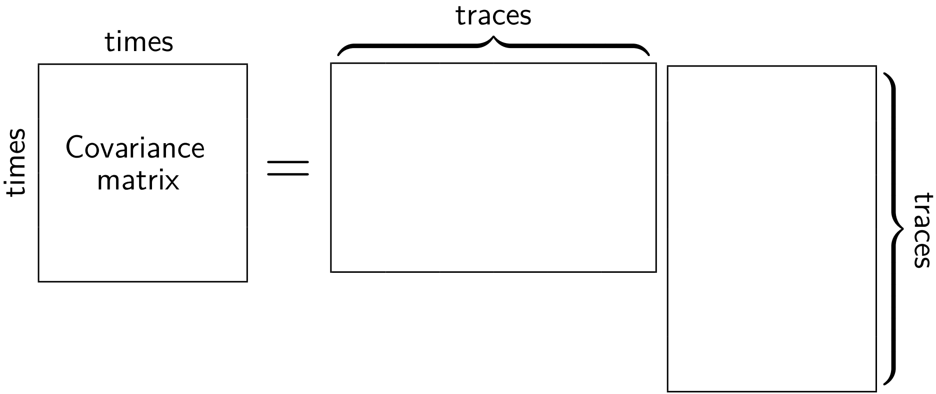


**Find:**

max variance directions

# Dimensionality Reduction

## Principal Component Analysis (PCA) - covariance matrix

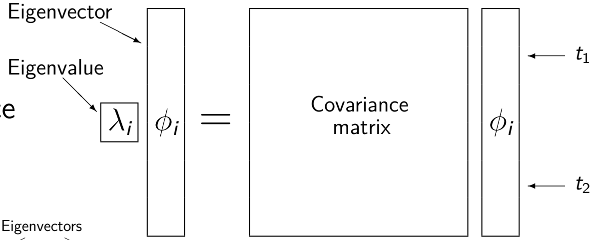


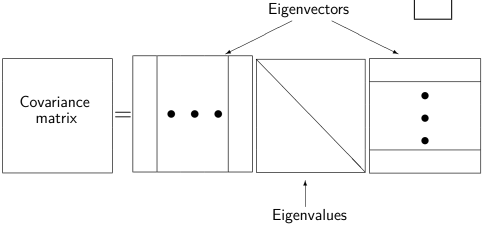

$$cov(t_1, t_2) \triangleq \sum_i trace_i[t_1] \cdot trace_i[t_2]$$

# Dimensionality Reduction

## Principal Component Analysis (PCA) - spectral theorem

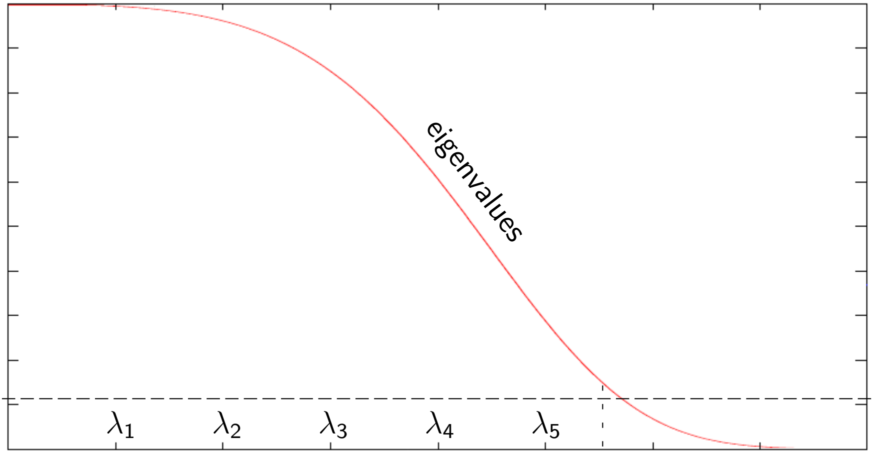
Spectral theorem applies to covariance matrices:




SVD (Singular Value Decomposition)

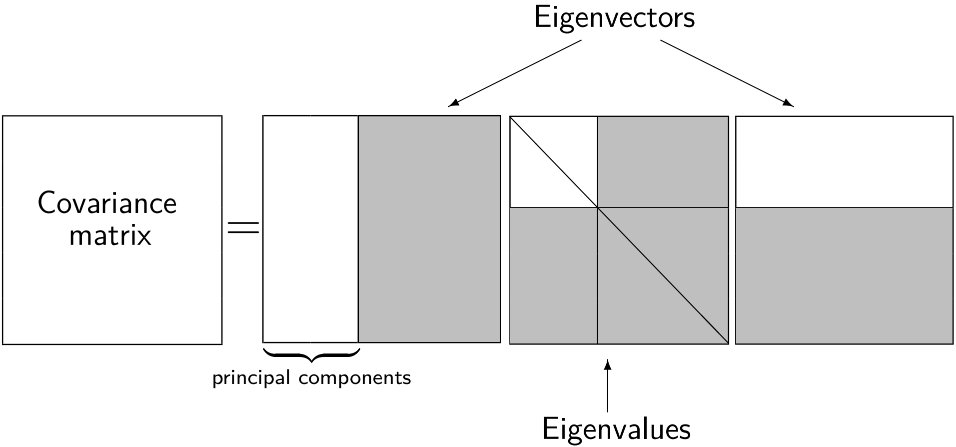
Spectral Theorem: $cov(t_1, t_2) = \sum_i \lambda_i \, \phi_i(t_1) \, \phi_i(t_2)$




Many datasets (incl. power traces) have a decaying cov. spectrum

# Dimensionality Reduction

## Principal Component Analysis (PCA) - truncated SVD

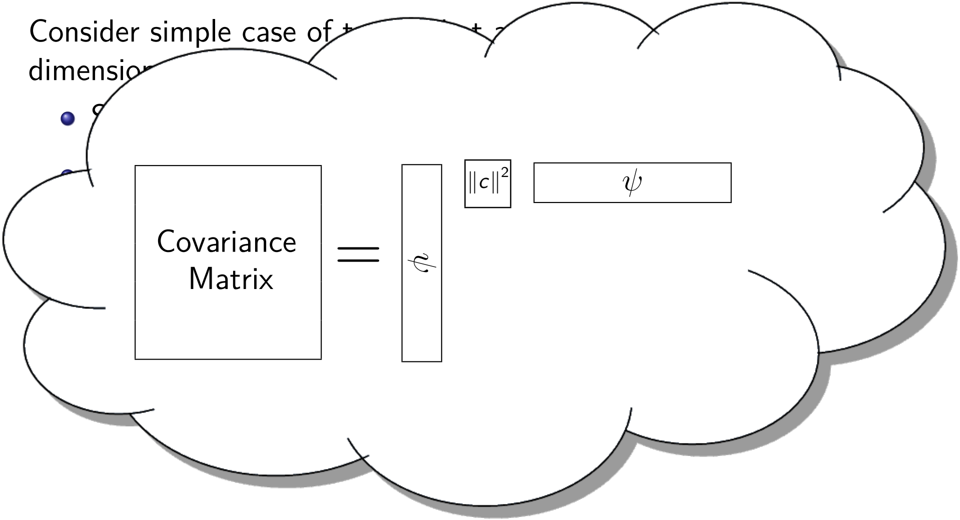



Approximate cov. matrix by truncating small eigenvalues from SVD

Consider simple case of traces that are all on the same high dimensional line

- Straight line is defined by a unit vector $\left\|\vec{\psi}\right\| = 1$
- Points on the line are defined by multiplying $\vec{\psi}$ by scalars
- The traces can be formulated as $x_i = c_i\vec{\psi}$
- Covariance: $\text{cov}(t_1, t_2) = \sum_i x_i[t_1]\, x_i[t_2] = \sum_i c_i\vec{\psi}[t_1]\, c_i\vec{\psi}[t_2] = (\sum_i c_i^2)\vec{\psi}[t_1]\vec{\psi}[t_2] = \|\vec{c}\|^2\, \vec{\psi}(t_1)\vec{\psi}(t_2) \qquad \vec{c} \triangleq (c_1, c_2, \ldots)$

Consider simple case of t
dimensio



Consider simple case of traces that are all on the same high dimensional line

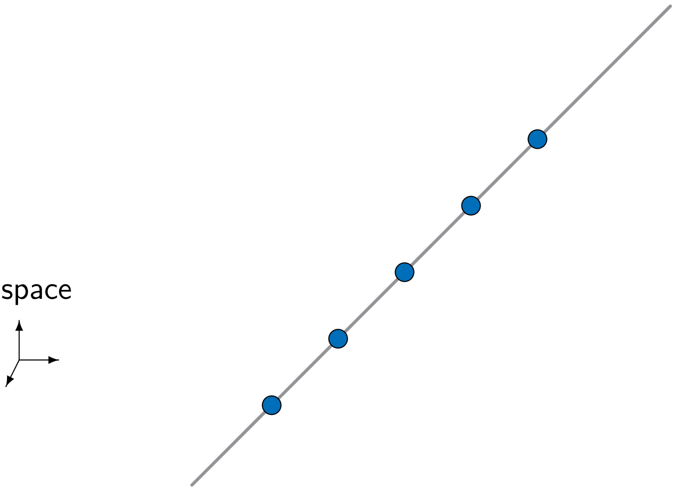
- Straight line is defined by a unit vector $\left\|\vec{\psi}\right\| = 1$
- Points on the line are defined by multiplying $\vec{\psi}$ by scalars
- The traces can be formulated as $x_i = c_i\vec{\psi}$
- Covariance: $\mathrm{cov}(t_1, t_2) = \sum_i x_i[t_1]\, x_i[t_2] = \sum_i c_i\vec{\psi}[t_1]\, c_i\vec{\psi}[t_2] = (\sum_i c_i^2)\vec{\psi}[t_1]\vec{\psi}[t_2] = \|\vec{c}\|^2\, \vec{\psi}(t_1)\vec{\psi}(t_2) \qquad \vec{c} \triangleq (c_1, c_2, \ldots)$

Covariance matrix has a single eigenvalue $\|\vec{c}\|^2$ and a single eigenvector $\vec{\psi}$, which defines principal direction of the trace-vectors

# Dimensionality Reduction

## Principal Component Analysis (PCA) - example

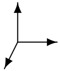
3D space

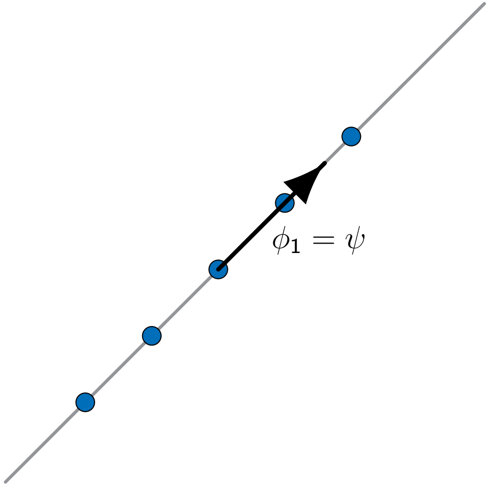


# Dimensionality Reduction

## Principal Component Analysis (PCA) - example

3D space

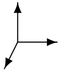


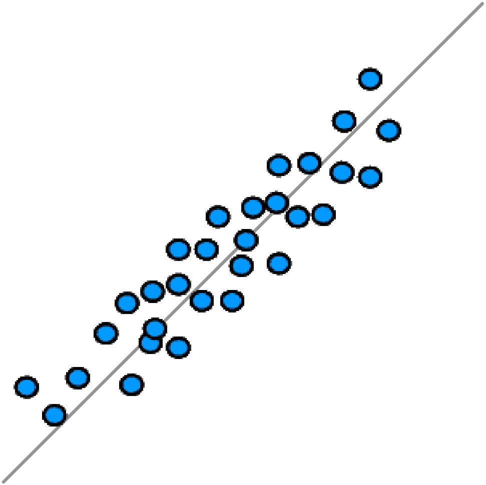


# Dimensionality Reduction

## Principal Component Analysis (PCA) - example

3D space



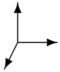


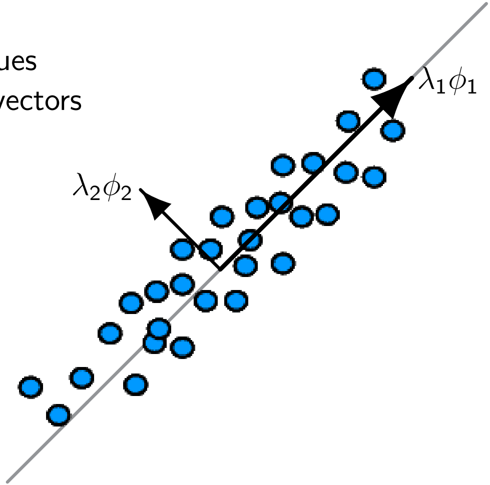
# Dimensionality Reduction

## Principal Component Analysis (PCA) - example

**Length**: eigenvalues
**Direction**: eigenvectors

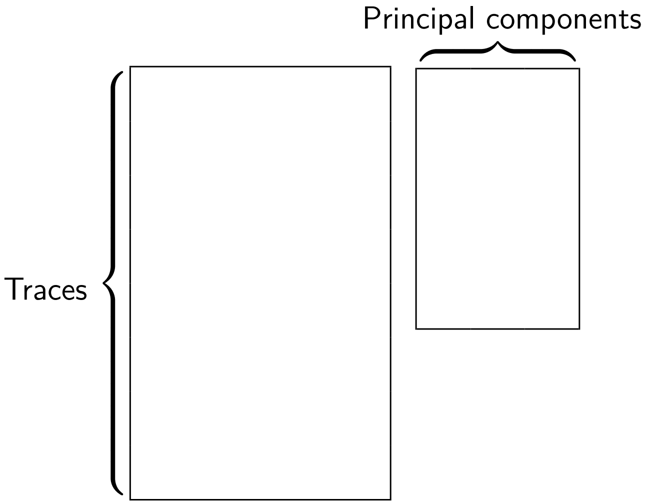
3D space



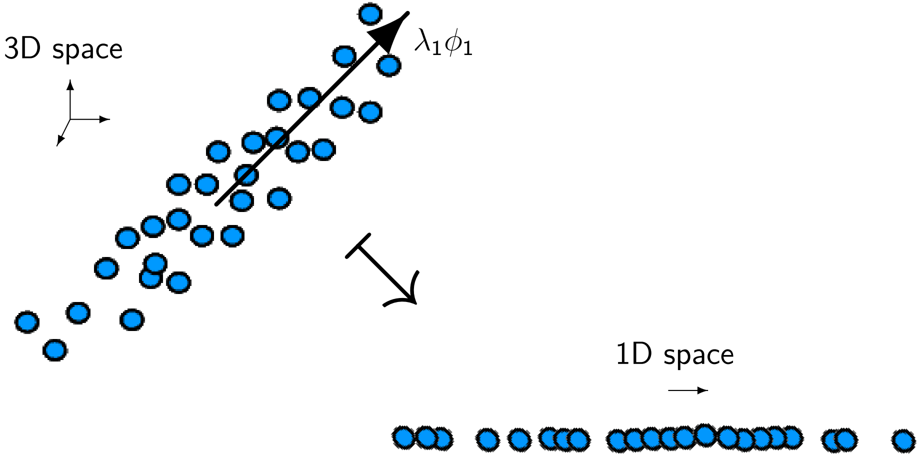



principal components $\Rightarrow$ max var directions

## Projection on principal components:



Projection on principal components:



## PCA algorithm:

1. Centering
2. Covariance
3. Eigendecomposition
4. Projection

**Alternative method**: Multi-Dimensional Scaling (MDS) - preserve distances/inner-products with minimal set of coordinates.

Short tutorial on PCA & MDS:

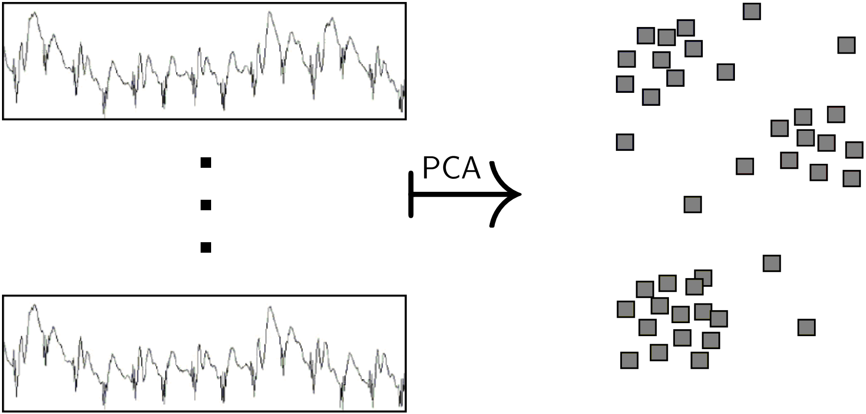
www.cs.haifa.ac.il/∼rita/uml_course/lectures/PCA_MDS.pdf

# Dimensionality Reduction

## Summary

Traces (high-dim. vectors):

Projected traces (low-dim. vectors):

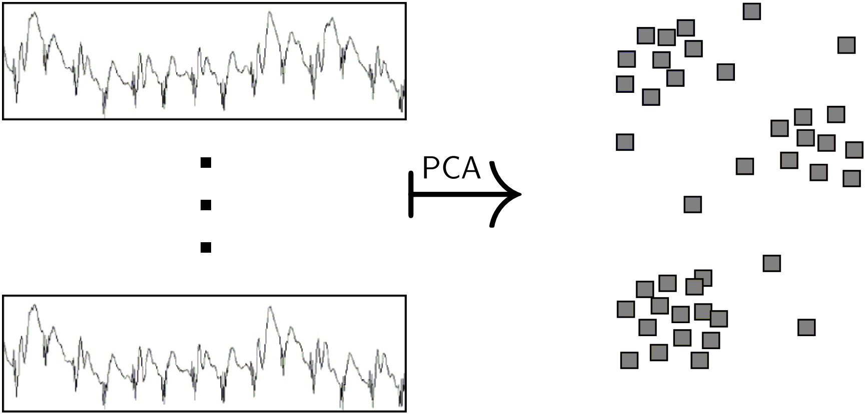


# Dimensionality Reduction

## Summary

Traces (high-dim. vectors):

Projected traces (low-dim. vectors):

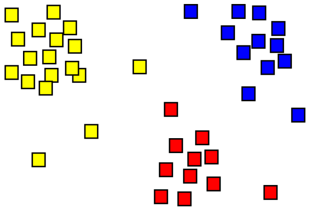



Next task: how to find keys from the low-dimensional vectors?

# Clustering & Classification
# with
# Support Vector Machine

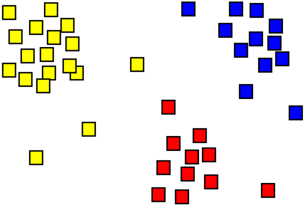
### Cluster analysis

*Clustering* - the task of grouping objects such that objects in the **same cluster** are more **similar** to each other than to those in other clusters.



### Cluster analysis

*Clustering* - the task of grouping objects such that objects in the **same cluster** are more **similar** to each other than to those in other clusters.



### Learning types

*Unsupervised learning*:
Trying to find hidden structures in unlabeled data.

*Supervised learning*:
Inferring functions from labeled training data.

# Classification

## Clustering & classification approaches

Classic TPA using Gaussian statistical models:

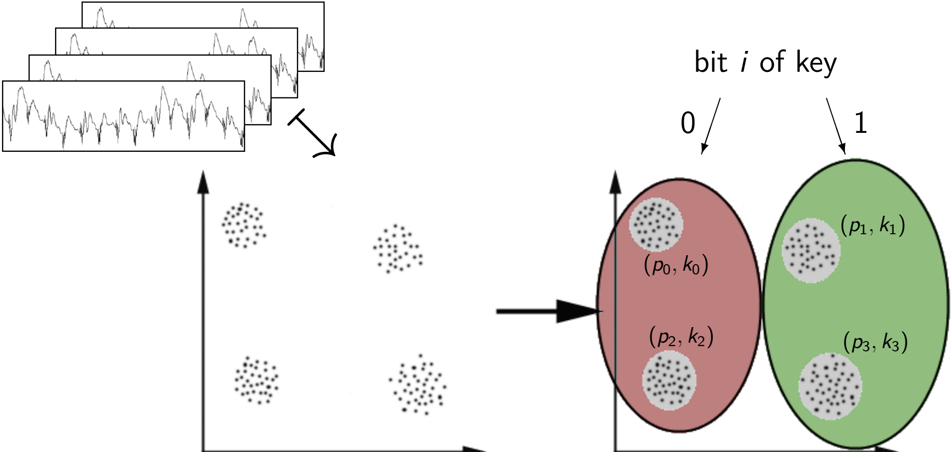
- The analysis considers many clusters (one for each key-plaintext pair)
- Clusters are assumed to look like normally distributed random variables $\vec{X}_{(k,p)} \sim \mathcal{N}(\vec{\mu}_{(k,p)}, \Sigma_{(k,p)})$
- Requires many traces for each key-plaintext pair to compute $\vec{\mu}_{(k,p)}$ & $\Sigma_{(k,p)}$

Simplified bit clustering with Support Vector Machine (SVM):

- Classify each bit separately - only two classes are considered for each bit
- Requires less training traces than classic TPA - traces are grouped by bit values, not by the key value
- No statistical assumptions required - geometric classification using a separating hyperplane

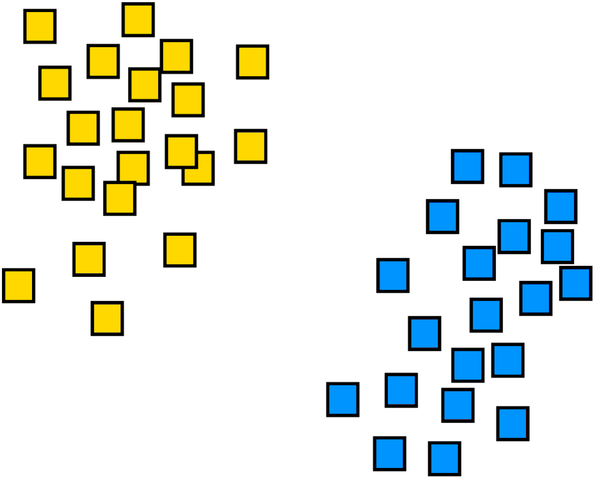
# Classification

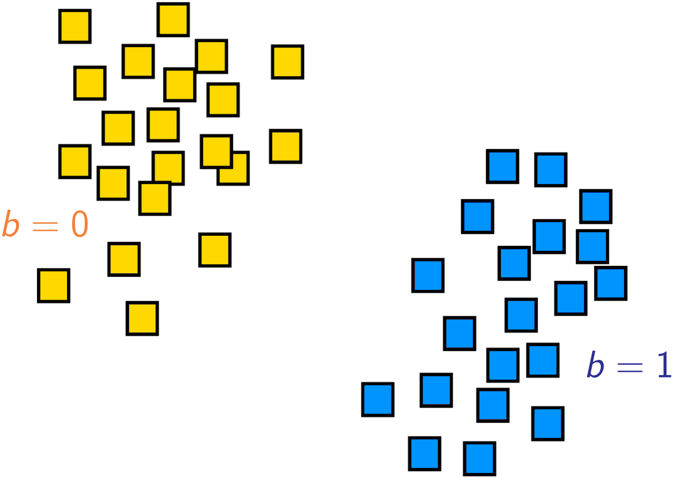
## Simplified bit clustering - only two classes



# Classification

## Support Vector Machine (SVM) - separation with hyperplane

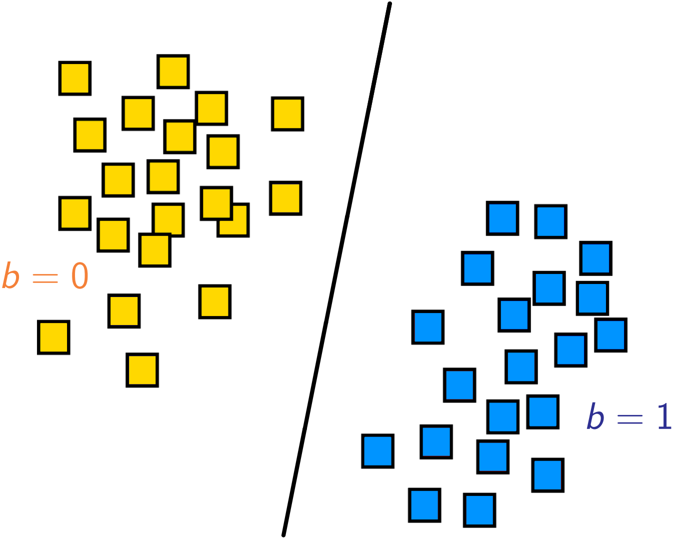




$b = 0$
$b = 1$

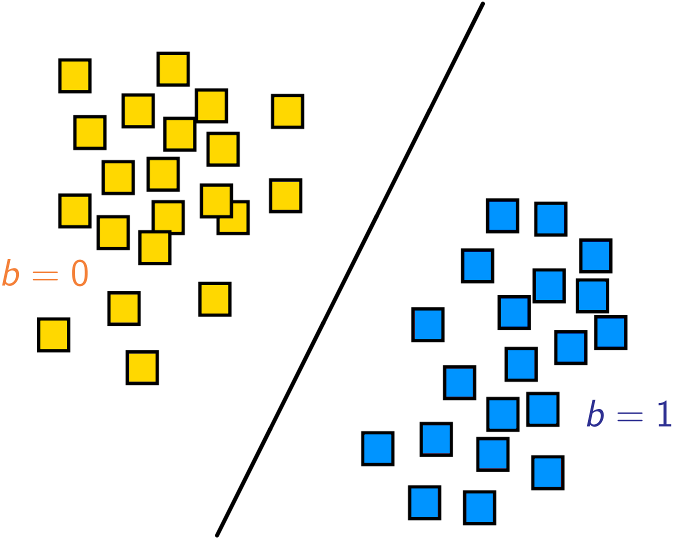
# Classification

## Support Vector Machine (SVM) - separation with hyperplane



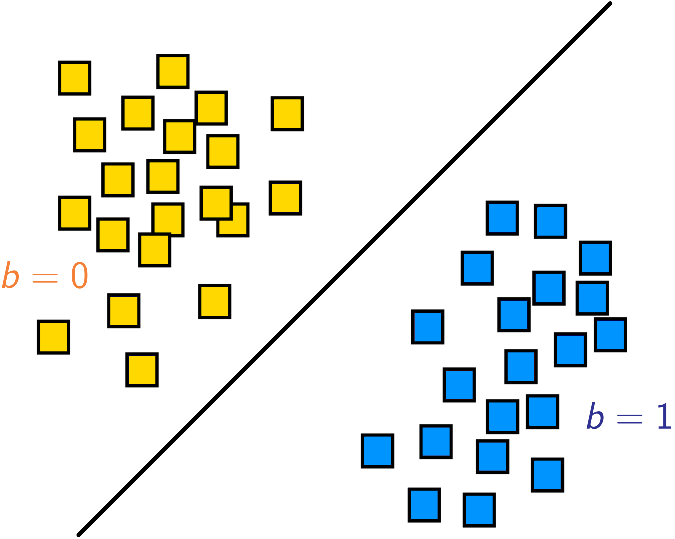
# Classification

## Support Vector Machine (SVM) - separation with hyperplane



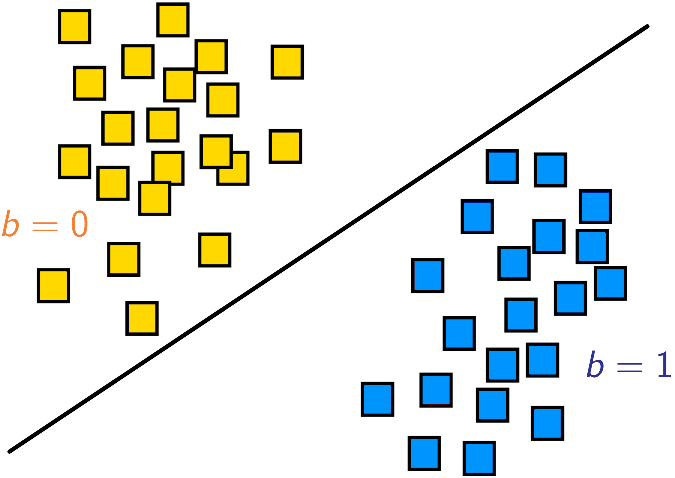
# Classification

## Support Vector Machine (SVM) - separation with hyperplane



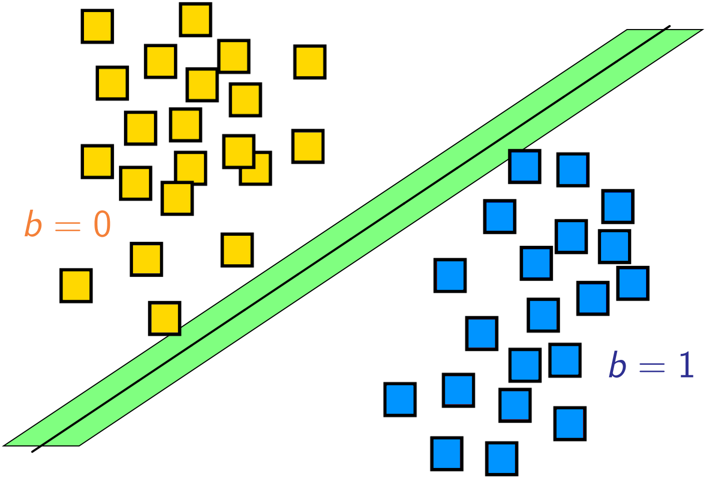
# Classification

## Support Vector Machine (SVM) - separation with hyperplane



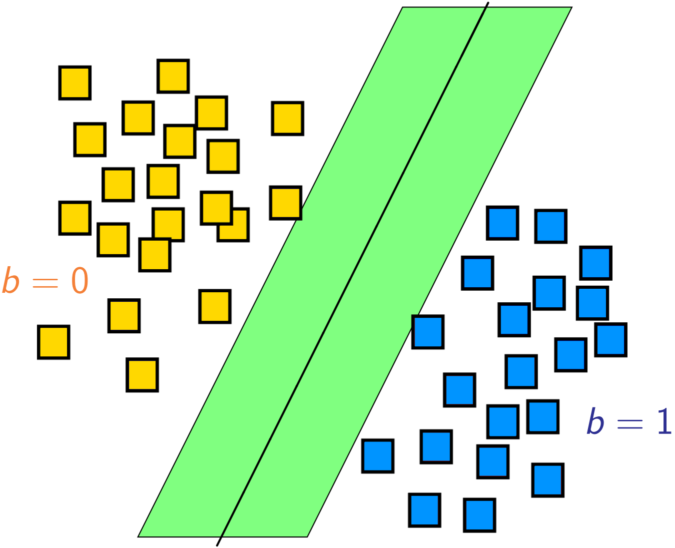
# Classification

## Support Vector Machine (SVM) - quantifying robustness with margins



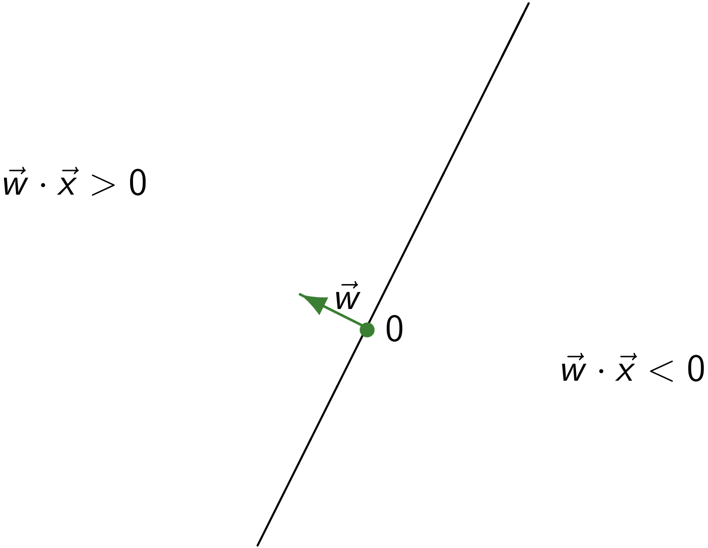
# Classification

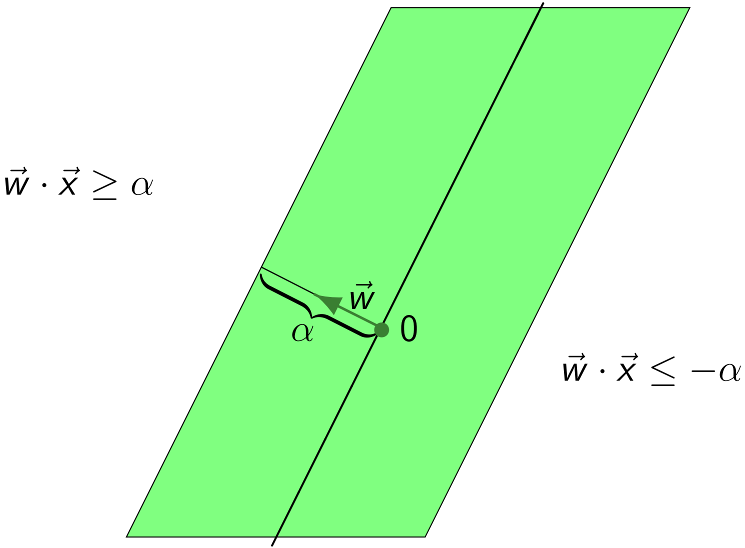
## Support Vector Machine (SVM) - quantifying robustness with margins



# Classification

## SVM formulation - hyperplane



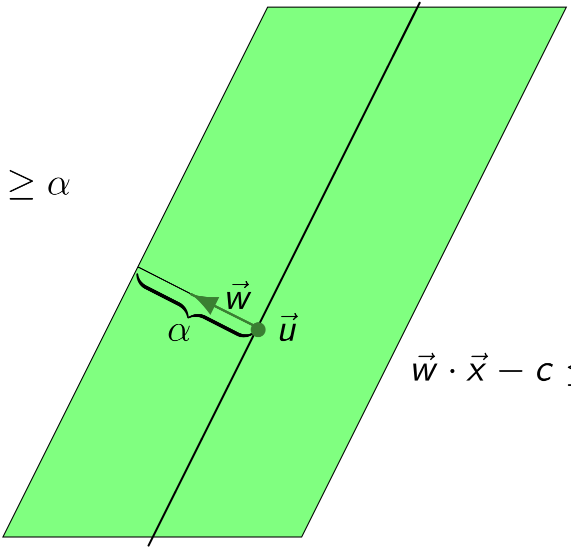

$\vec{w} \cdot \vec{x} \geq \alpha$
$\vec{w}$
$\alpha$
0
$\vec{w} \cdot \vec{x} \leq -\alpha$

# Classification

## SVM formulation - shifted hyperplane with margin

$$\boxed{c = \vec{w} \cdot \vec{u}}$$

$$\vec{w} \cdot \vec{x} - c \geq \alpha$$




$$\vec{w} \cdot \vec{x} - c \leq -\alpha$$

## SVM training

Input:

- Points $\{\vec{x}_i\}$ from PCA of the traces
- Labels $\{b_i\}$ according to attacked bit:

$$b_i = \begin{cases} 1 & \text{bit is } 0 \\ -1 & \text{bit is } 1 \end{cases}$$

Solve the quadratic program (e.g., using Lagrange multipliers):

$$\begin{array}{ll} \text{Find} & \max \alpha \\ \text{s.t.} & \vec{w} \cdot \vec{x}_i - c \geq b_i \alpha \end{array}$$

Output: the solution $(\vec{w}, c, \alpha)$

# Classification

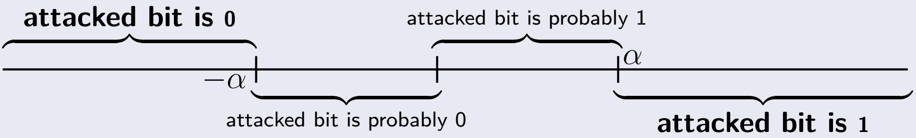
## SVM algorithm

### SVM classifier

Input:

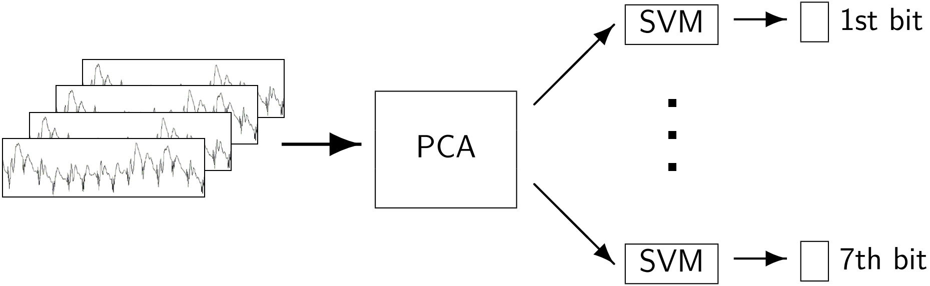
- New point $\vec{x}$ from PCA projection of attacked trace
- The solution $(\vec{w}, c, \alpha)$ from SVM training

Classify by value of $\vec{w} \cdot \vec{x} - c$:



# Power Analysis with PCA & SVM

based on recent paper[2]




Use single PCA & multiple SVMs (one per bit)

to learn traces (in training phase) and attack a key byte

[2] "Side channel attack: an approach based on machine learning", 2011, by L. Lerman, G. Bontempi, and O. Markowitch

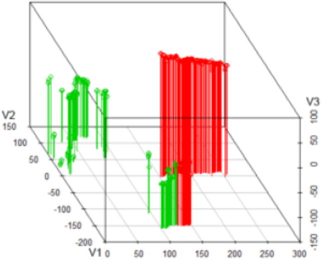
# Power Analysis with PCA & SVM

## based on recent paper[2]

PCA results (colored by specified bit values):

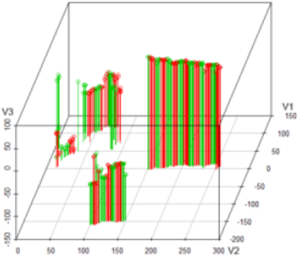
bit = 1   bit = 0

Effective SVM attack:




7th bit

SVM attack will fail:




3rd bit

Empirical results (on 3DES): desc. success rate (by bit position)

7th bit success rate: ∼95% ⟶ 1st bit success rate: ∼50%

[2] "Side channel attack: an approach based on machine learning", 2011, by L. Lerman, G. Bontempi, and O. Markowitch

# Power Trace Alignment

# Power Trace Alignment

## Motivation

Power traces can be misaligned for several reasons, such as

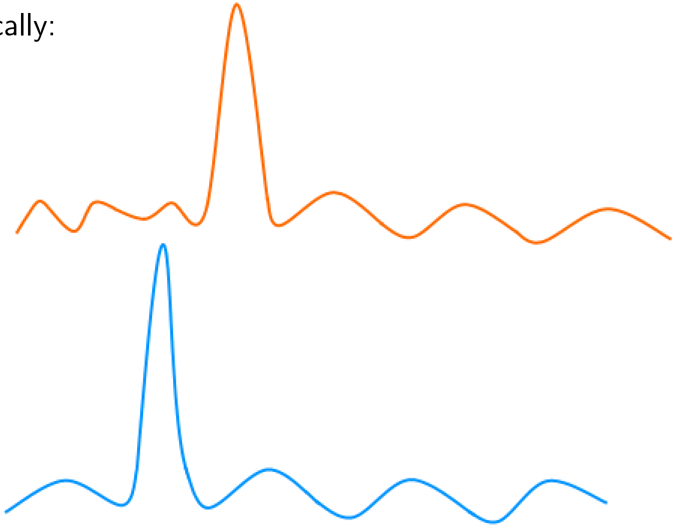
- Synchronization issues between the sampling devices and the tested hardware
- Clock variabilities and instabilities
- Intentional countermeasures such as delays and modulations

Misaligned traces $\Rightarrow$ incorrect/inaccurate correlations $\Rightarrow$ wrong classification and useless attacks

# Power Trace Alignment

## Naïve approach: static alignment by time offset

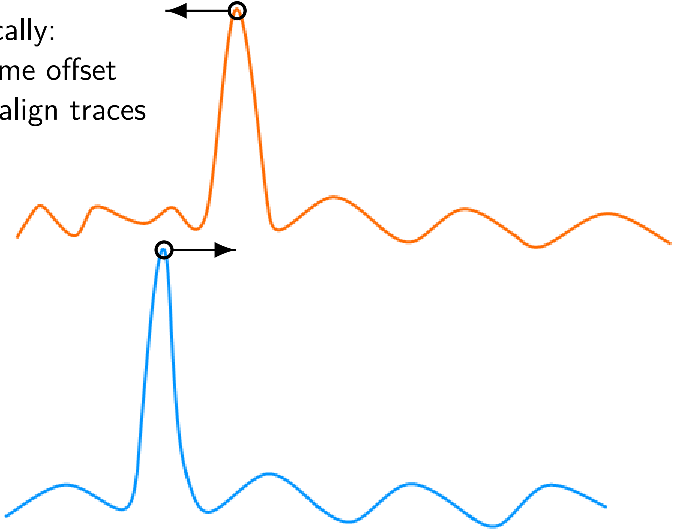
Theoretically:



# Power Trace Alignment

## Naïve approach: static alignment by time offset

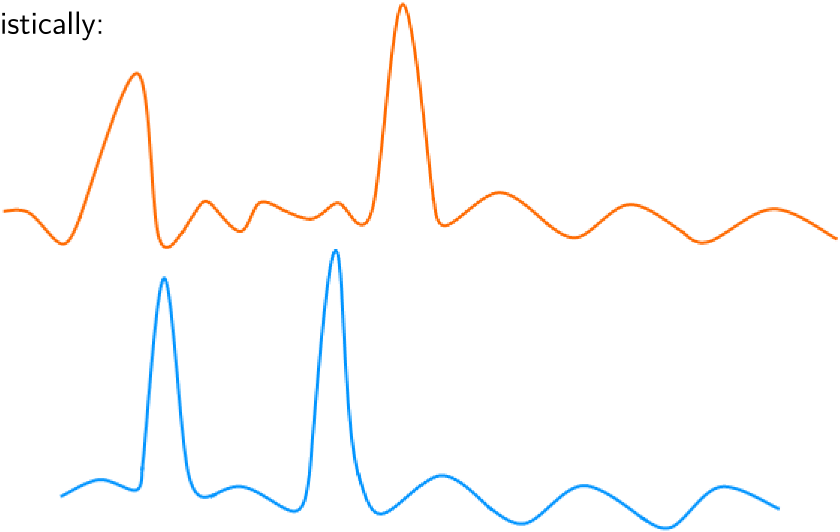
Theoretically:
Use time offset
to align traces



# Power Trace Alignment

Naïve approach: static alignment by time offset

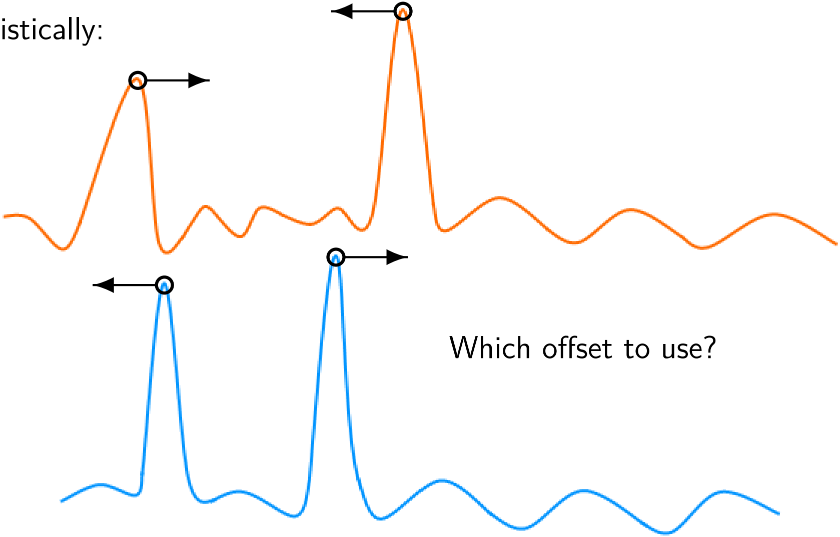
Realistically:



# Power Trace Alignment

## Naïve approach: static alignment by time offset

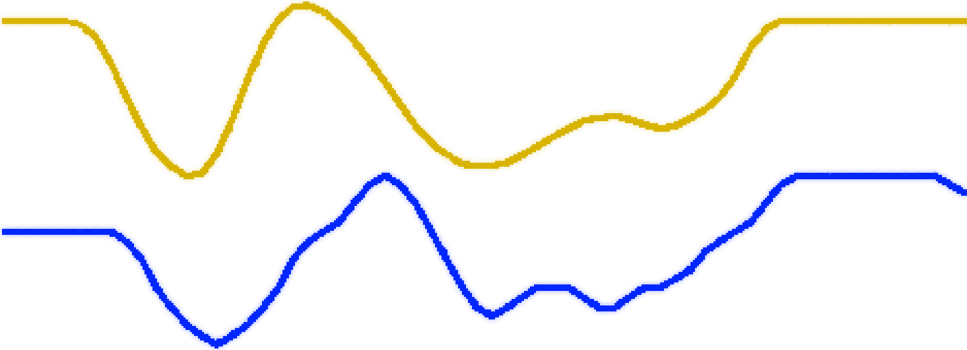
Realistically:



Which offset to use?

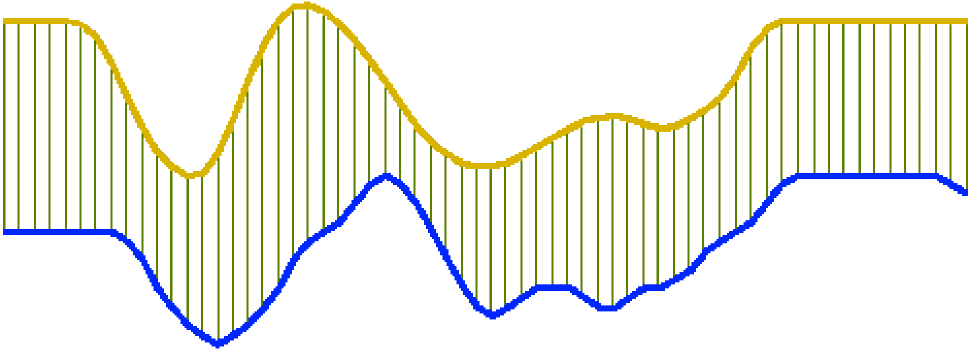
# Power Trace Alignment

## Machine-learning approach: adaptive alignment by **Dynamic Time Warp (DTW)**



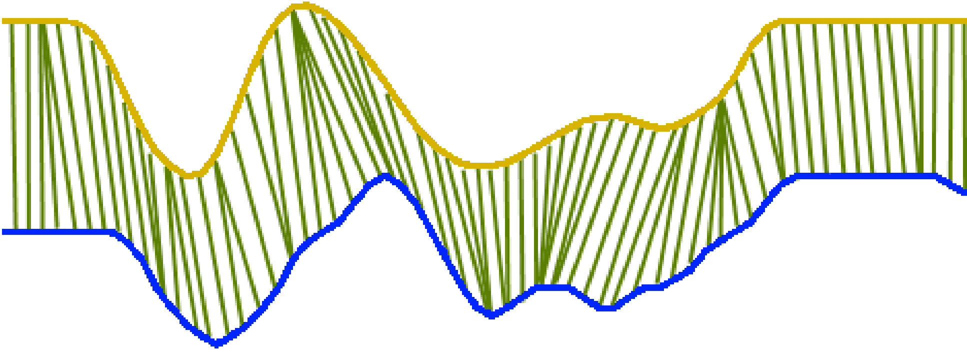
# Power Trace Alignment

## Machine-learning approach: adaptive alignment by **Dynamic Time Warp (DTW)**



# Power Trace Alignment

Machine-learning approach: adaptive alignment by **Dynamic Time Warp (DTW)**



# Power Trace Alignment

## Using pairwise alignment in an attack

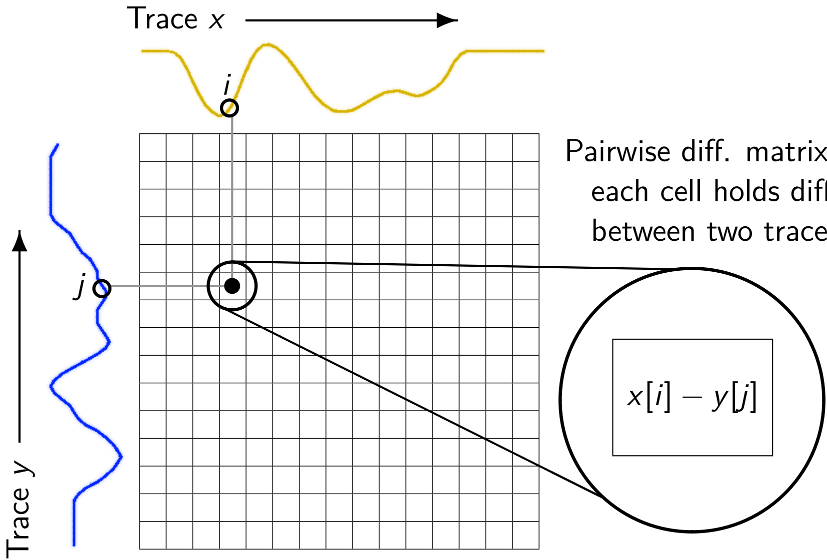
### Training:

1. Acquire power traces
2. Choose reference trace (e.g., arbitrarily or use mean of all traces)
3. Align each trace to the reference trace using the pairwise alignment
4. Apply training algorithm (e.g., PCA & SVM) to the aligned traces

### Online:

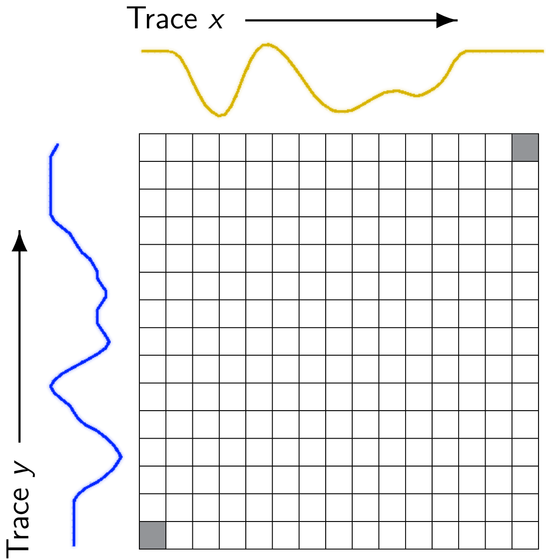
1. Acquire trace from attacked hardware
2. Align trace to the reference trace (from the training) using pairwise alignment
3. Apply classification algorithm (e.g., PCA & SVM)

# Pairwise Power Trace Alignment




Pairwise diff. matrix:
each cell holds difference
between two trace entries

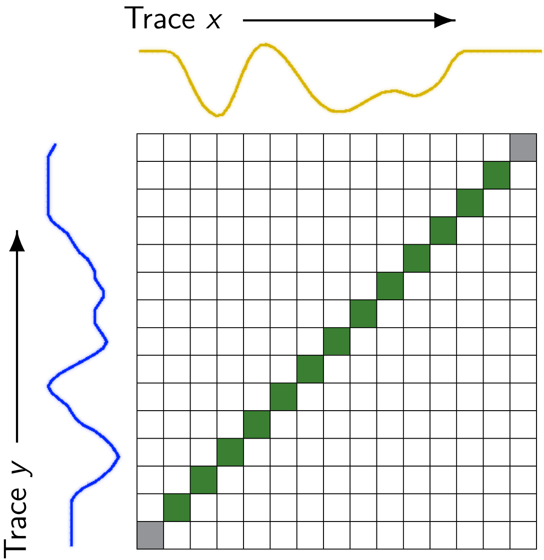
# Pairwise Power Trace Alignment




Alignment path:

get from start to end
of both traces

# Pairwise Power Trace Alignment



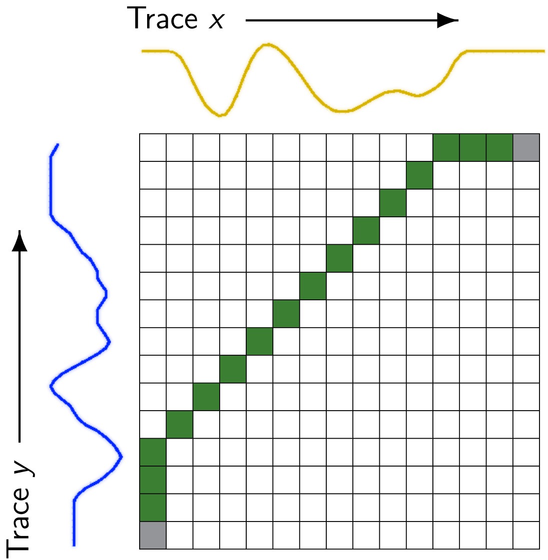

1:1 alignment:

trivial - nothing modified by the alignment

Aligned distance:

$$\sum \left(\blacksquare\right)^2 = \|x - y\|^2$$

# Pairwise Power Trace Alignment



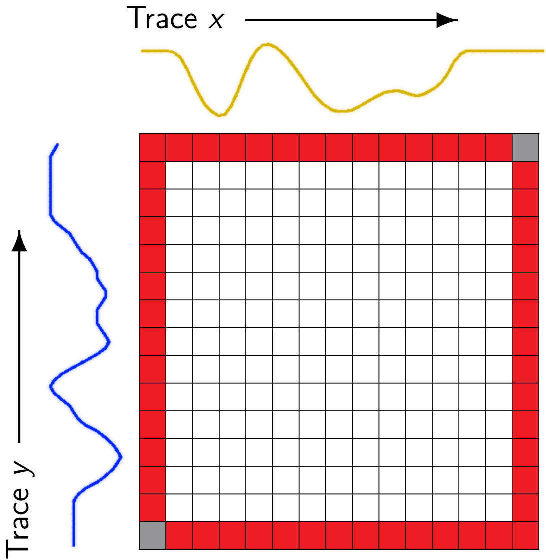

Time offset:

works sometimes, but
not always optimal

Aligned distance:

$$\sum \left( \blacksquare \right)^2 = ?$$

# Pairwise Power Trace Alignment



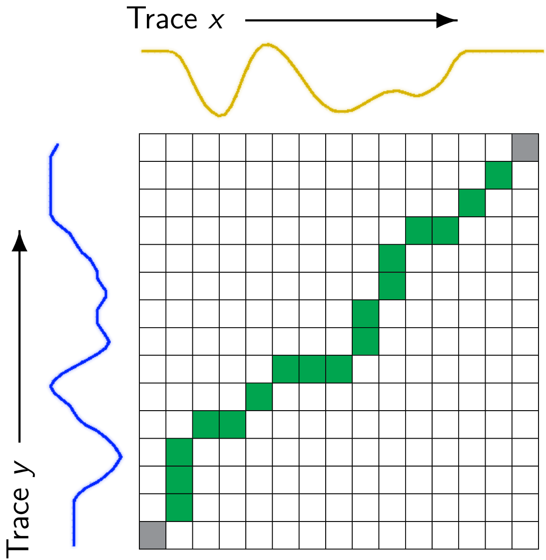

Extreme offset:

complete misalignment - worst alignment alternative

Aligned distance:

$$\sum \left(\blacksquare\right)^2 = \|x\|^2 + \|y\|^2$$

# Pairwise Power Trace Alignment



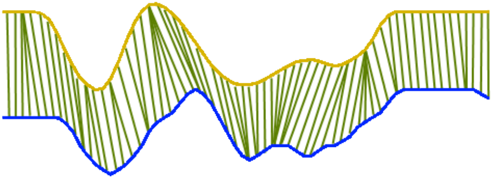

Optimal alignment:
Optimize alignment by minimizing aligned distance

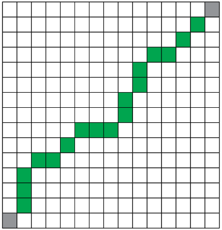
Aligned distance:

$$\sum \left( \blacksquare \right)^2 = \min$$

# Power Trace Alignment

## Finding optimal pairwise alignment





### Dynamic Programming

- A method for solving complex problems by breaking them down into simpler subproblems.
- Applicable to problems exhibiting the properties of overlapping subproblems and optimal substructure.
- Better performances than naive methods that do not utilize the subproblem overlap.

# Power Trace Alignment

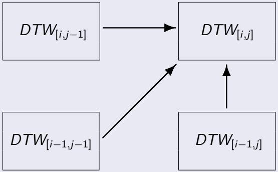
## Dynamic Time Warp (DTW)

### Basic DTW Algorithm:

For each trace-time $i$ and for each trace-time $j$:

- Set $cost \leftarrow (x[i] - y[j])^2$
- Set the optimal distance at stage $[i, j]$ to:

$$DTW_{[i,j]} \leftarrow cost + \min \left\{ \begin{array}{c} DTW_{[i,j-1]} \\ DTW_{[i-1,j-1]} \\ DTW_{[i-1,j]} \end{array} \right\}$$



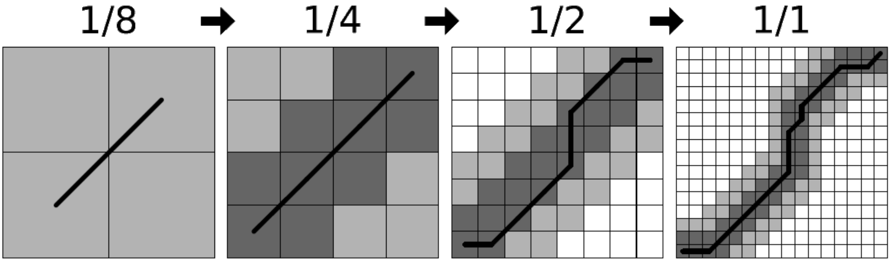

Optimal distance: $DTW_{[m,n]}$ (where $m$ & $n$ are lengths of traces).

Optimal alignment: backtracking the path leading to $DTW_{[m,n]}$ via min-cost choices of the algorithm

# Power Analysis with DTW-based Alignment

based on recent paper[3]

Use coarse-grained matrices to avoid bad/unreasonable portions:




Drill down by fine graining to approximate the optimal alignment with quasi-linear time & space requirements

[3] "Improving Differential Power Analysis by Elastic Alignment", 2011, by J.G.J. van Woudenberg, M.F. Witteman, and B. Bakker

# Power Analysis with DTW-based Alignment

based on recent paper[3]

## Experimental results:

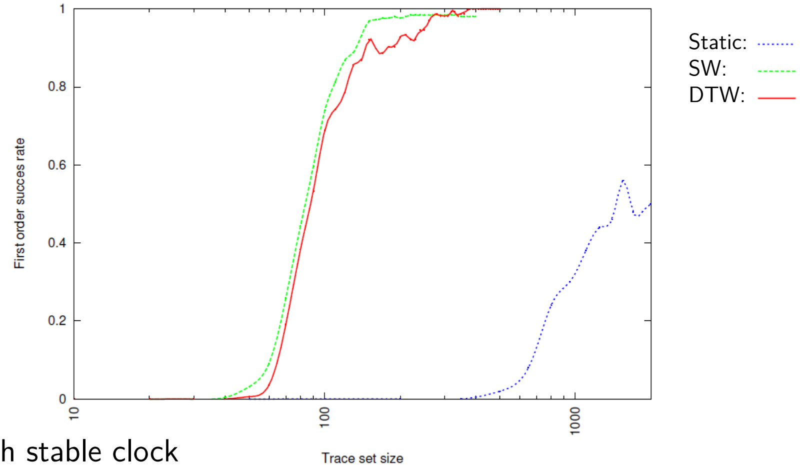
Compare correlation DPA using 3 alignment methods:

- **Static:** Simple static alignment by time offset
- **SW:** Replace trace entries with avg. of sliding window
  - Not strictly an alignment method, but simple & sometimes effective
- **DTW:** Elastic alignment with DTW

[3] "Improving Differential Power Analysis by Elastic Alignment", 2011, by J.G.J. van Woudenberg, M.F. Witteman, and B. Bakker

# Power Analysis with DTW-based Alignment

based on recent paper[3]

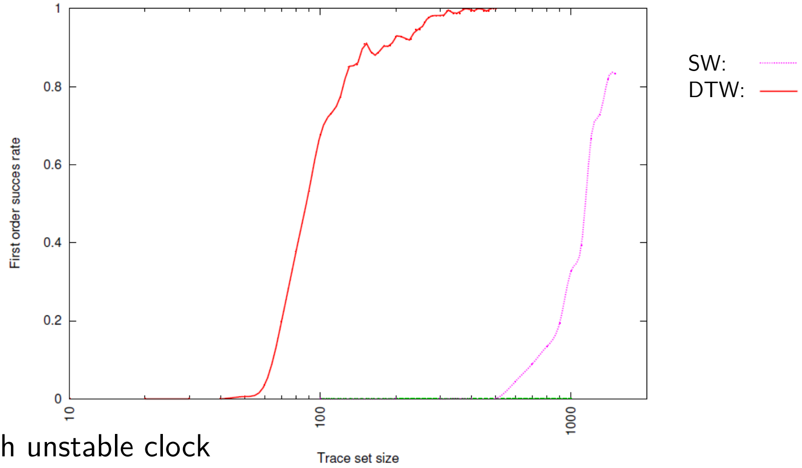



DES with stable clock

[3] "Improving Differential Power Analysis by Elastic Alignment", 2011, by J.G.J. van Woudenberg, M.F. Witteman, and B. Bakker

# Power Analysis with DTW-based Alignment

based on recent paper[3]




DES with unstable clock

[3]"Improving Differential Power Analysis by Elastic Alignment", 2011, by J.G.J. van Woudenberg, M.F. Witteman, and B. Bakker

# Analyzing Non-Cryptographic Leaks with Hidden Markov Model

# Information & Activity Leaks

Consider secret sequence of activities and leaked information with the following properties:

- Contains information about the secret sequence
- Contains noise
- Insufficient for directly recovering the secret information

If activities follow **known statistical patterns**, then an attacker can **“guess” secret sequence** from noisy leaks.

Attack: find best hypothesis such that:

1. It matches the leaked data
2. Has high probability according to statistical distribution of activity sequences

# Information & Activity Leaks

## Can it work?

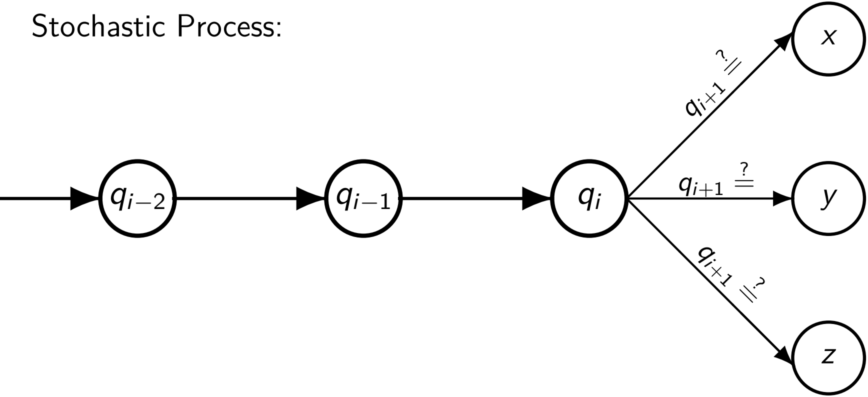
Leaked information can be used for more than cryptographic purposes:

- Users are predictable - most activities are similar & repetitive
  - Internet - common websites and surfing routines
  - Emails/documents - linguistic models
  - Passwords - most common password is "password"
    - Others examples: "querty", "letmein", "trustno1", "dragon", "monkey", "ninja", and "jesus".
    - News services often publish lists of most common passwords of the year/month
- Guess activities/information by detecting "reasonable" usage patterns from leaked data

A statistical model of user activity profile can be used for this task.

# Markov Chain

Stochastic Process:

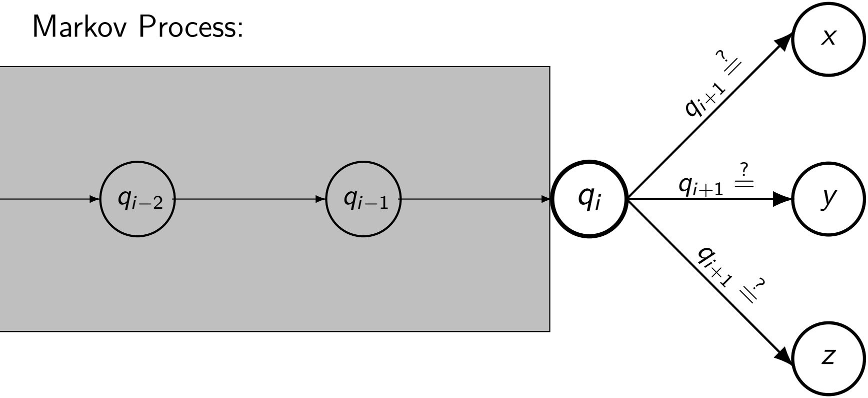



**Transition probabilities:**

$$\Pr\left[q_{i+1} =? \mid q_i, q_{i-1}, \ldots, q_2, q_1\right]$$

# Markov Chain

Markov Process:




**Transition probabilities (no history):**

$$\Pr\left[q_{i+1} = ? \mid q_i\right] = \Pr\left[q_{i+1} = ? \mid q_i, q_{i-1}, \ldots, q_2, q_1\right]$$


esc
! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 ( 9 ) 0 delete
tab Q W E R T Y U I O P
caps lock A S D F G H J K L enter return
shift Z X C V B N M < , > . ? / shift
fn ctrl alt option cmd ⌘ ⌘ cmd alt option

# Markov Chain

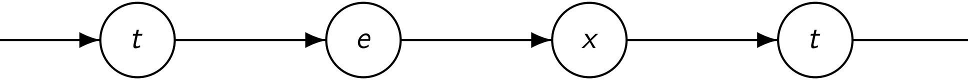
## Keyboard structure & text auto-complete

# Markov Chain

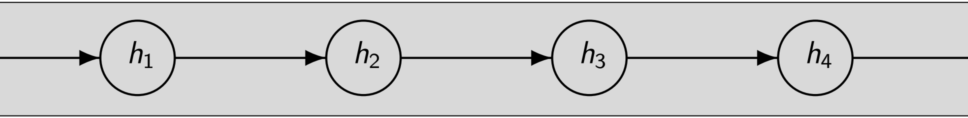
## Keyboard structure & text auto-complete

# Markov Chain

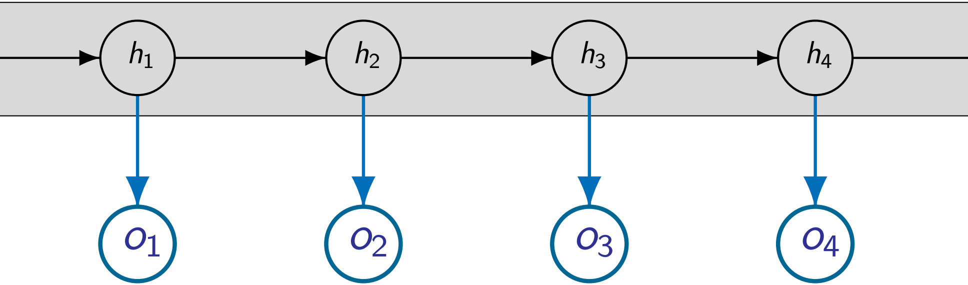
## Keyboard structure & text auto-complete

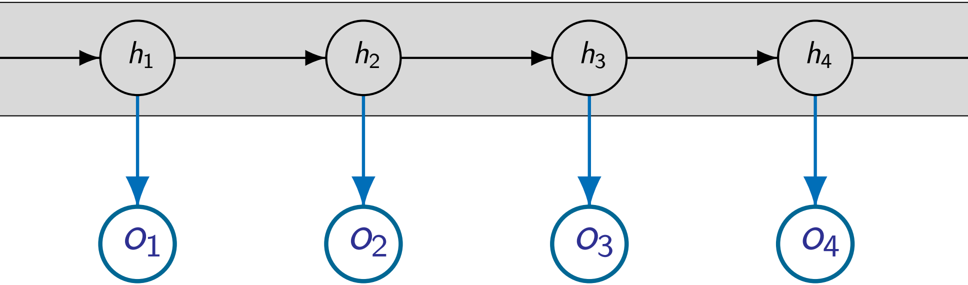
# Hidden Markov Model



# Hidden Markov Model



# Hidden Markov Model



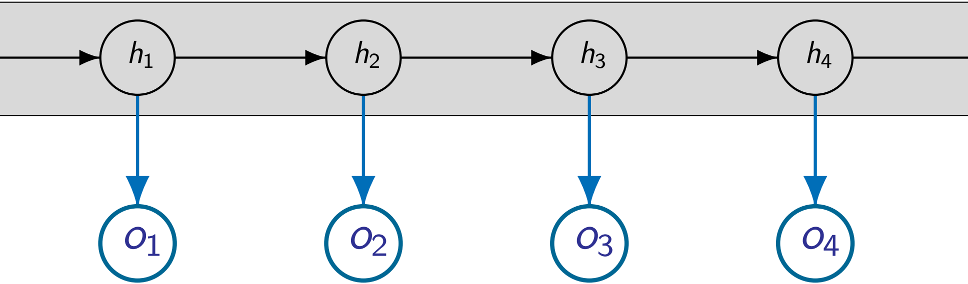

**Transition probabilities:**

$$\Pr\left[h_{i+1} = ? \mid h_i\right]$$

**Leak probabilities:**

$$\Pr\left[o_i = ? \mid h_i\right]$$

# Hidden Markov Model

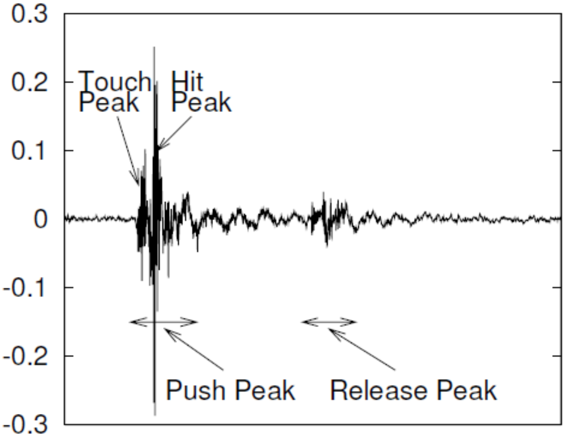



## Viterbi Algorithm

A dynamic programming algorithm for finding the most likely sequence of hidden states, especially in the context of Hidden Markov models.

# Acoustic Analysis of Keyboards

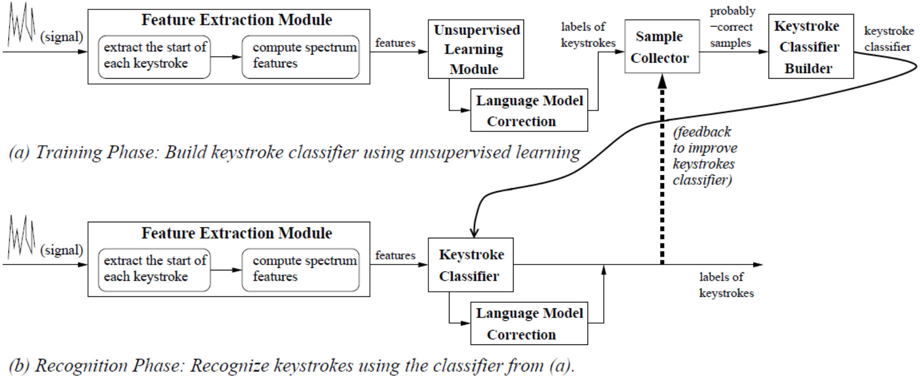
based on paper[4]




[4] "Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

# Acoustic Analysis of Keyboards

## based on paper[4]




[4] "Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

# Acoustic Analysis of Keyboards

based on paper[4]

## Typed text:

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

[4]"Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

# Acoustic Analysis of Keyboards

based on paper[4]

## HMM only:

```
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 acyivitt, srem the
grosth of small arrists and lead to lost
cobs and dimished sales tas revenue.
```

[4] "Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

# Acoustic Analysis of Keyboards

based on paper[4]

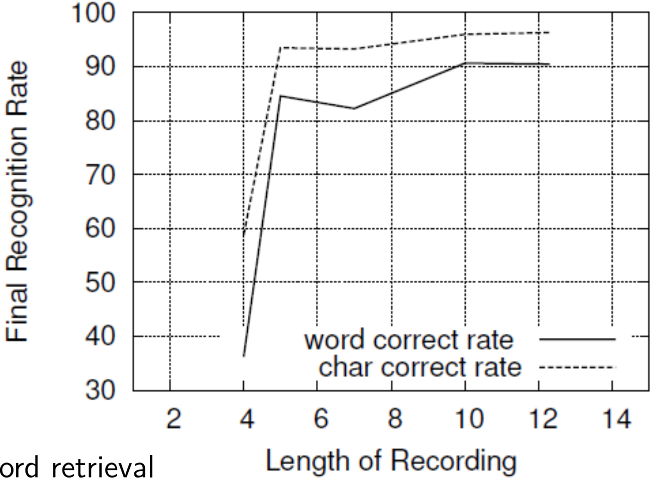
## HMM & spelling corrections:

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.

[4] "Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

# Acoustic Analysis of Keyboards

based on paper[4]

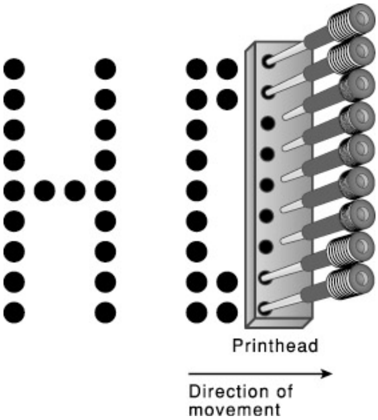



Password retrieval

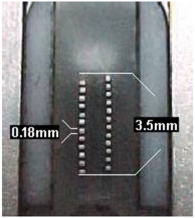
[4] "Keyboard Acoustic Emanations Revisited",2005, by L. Zhuang, F. Zhou, and J.D. Tygar

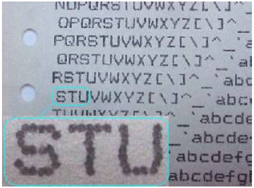
# Acoustic Analysis of Printers

## based on paper[5]




(Picture taken from URL:flylib.com/books/en/2.374.1.27/1/)



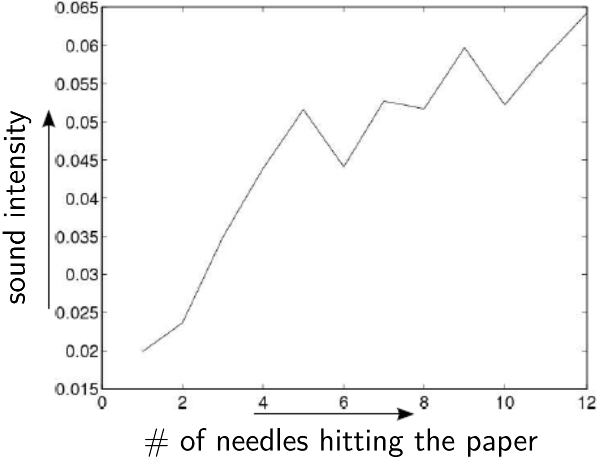




(Pictures taken from URL:mindmachine.co.uk/book/print_06_dotmatrix_overview01.html)

[5] "Acoustic Side-Channel Attacks on Printers",2010, by M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder

# Acoustic Analysis of Printers

based on paper[5]




[5] "Acoustic Side-Channel Attacks on Printers",2010, by M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder

# Acoustic Analysis of Printers

## based on paper[5]

### Training:

- Feature extraction (split into words, noise reduction, etc.)
- Construct DB with $(\text{word}, \text{sound})$ pairs

### Online:

- Feature extraction (same as in training)
- For each word:
  - Sort DB by similarity/difference from recorded sound
  - Reorder DB by n-gram/word distribution using HMM
  - Guess printed word as the top candidate from reordered DB

[5] "Acoustic Side-Channel Attacks on Printers",2010, by M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder

# Acoustic Analysis of Printers

## based on paper[5]

| | Text 1 | Text 2 | Text 3 | Text 4 | **Overall** |
|---|---|---|---|---|---|
| Basic Top 1 (*Top 3*) | 60.5 % (*75.1 %*) | 66.5 % (*79.2 %*) | 62.8 % (*78.7 %*) | 61.5 % (*77.9 %*) | **62.9 %** (*78.0 %*) |
| HMM 3-gram | 66.7 % | 71.8 % | 71.2 % | 69.0 % | **69.9 %** |

| | Declaration 1 | Declaration 2 |
|---|---|---|
| Basic Top 1 (*Top 3*) | 59.5 % (*77.8 %*) | 57.5 % (*72.6 %*) |
| HMM 3-gram (using general-purpose corpus) | 68.3 % | 60.8 % |
| **HMM 3-gram (using domain-specific corpus)** | **95.2 %** | **72.5 %** |

[5] "Acoustic Side-Channel Attacks on Printers",2010, by M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder

# Other Activity Leaks

Other activity leaks to which machine learning (and other tools) are applied:

- Offensively:
  - Other cases of trace analysis (e.g., frequency domain)
  - Traffic analysis
  - Deanonymization
- Defensively:
  - Authentication
  - Malware code detection
  - Malware command-and-control traffic detection
  - DDoS detection

# Further Reading I

## Side-channel attacks using machine learning tools:

- "Further hidden Markov model cryptanalysis" (2005) by P.J. Green, R. Noad, N.P. Smart
- "Analyzing side channel leakage of masked implementations with stochastic methods" (2007) by K. Lemke-Rust & C. Paar
- "Side channel attacks on cryptographic devices as a classification problem" (2007) by P. Karsmakers, B. Gierlichs, K. Pelckmans, K. De Cock, J. Suykens, B. Preneel, B. De Moor
- "Theoretical and practical aspects of mutual information based side channel analysis" (2009) by E. Prouff & M. Rivain
- "Cache-timing template attacks" (2009) by B.B. Brumley & R.M. Hakala
- "Machine learning in side-channel analysis: a first study" (2011) by G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, J. Vandewalle
- *"Side channel attack: an approach based on machine learning" (2011) by L. Lerman, G. Bontempi, O. Markowitch*
- "Side channel cryptanalysis using machine learning" (2012) by H. He, J. Jaffe, & L. Zou
- "PCA, eigenvector localization and clustering for side-channel attacks on cryptographic hardware devices" (2012) by D. Mavroeidis, L. Batina, T. van Laarhoven, E. Marchiori
- "Efficient Template Attacks Based on Probabilistic Multi-class Support Vector Machines" (2013) by T. Bartkewitz & K. Lemke-Rust

# Further Reading II

## Trace alignment:

- "Recovering secret keys from weak side channel traces of differing lengths" (2008) by C.D. Walter
- "Side Channel Analysis enhancement: a proposition for measurements resynchronisation" (2011) N. Debande, Y. Souissi, M. Nassar, S. Guilley, T.H. Le, J.L. DangerBakker
- *"Improving differential power analysis by elastic alignment" (2011) by J.G.J. van Woudenberg, M.F. Witteman, B. Bakker*
- "A general approach to power trace alignment for the assessment of side-channel resistance of hardened cryptosystems" (2012) by Q. Tian & S.A. Huss

## Information retrieval from leaked data:

- *"Keyboard acoustic emanations revisited" (2005) by L. Zhuang, F. Zhou, J.D. Tygar*
- *"Acoustic side-channel attacks on printers" (2010) by M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder*
- "Building a side channel based disassembler" (2010) by T. Eisenbarth, C. Paar, Björn Weghenkel
- "Automated black-box detection of side-channel vulnerabilities in web applications" (2011) by P. Chapman & D. Evans
- "Current events: identifying webpages by tapping the electrical outlet" (2012) by S. S. Clark, B. A. Ransford, J. M. Sorber, W. Xu, E. G. Learned-Miller, K. Fu
- "Engineering statistical behaviors for attacking and defending covert channels" (2013) by V. Crespi, G. Cybenko, A. Giani

# Conclusion

## Machine learning

Retrieve meaningful information from vast amounts of leaked data.

## Machine learning tools/concepts:

- Training/testing scheme
- Dimensionality reduction with PCA
- Clustering/classification with SVM
- Alignment with DTW
- Predicting/guessing usage patterns with HMM

## Side channel applications

- Template based power analysis & power trace alignment
- Acoustic analysis of computer peripherals