Robust Profiling for DPA-Style Attacks

Carolyn Whitnall[1], Elisabeth Oswald[1]

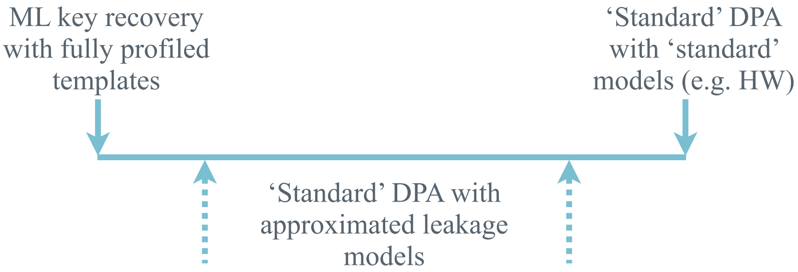
[1]Department of Computer Science, University of Bristol

carolyn.whitnall@bris.ac.uk

September 2015

Introduction

Top line: Extracting ‘portable’ power models for DPA attacks.



Outline:

- Preliminaries: ‘Standard’ DPA; different ‘types’ of power model; unsupervised (k-means) clustering.
- Proposed methodology: unsupervised clustering for building nominal power models.
- Experimental results.

Top line: Extracting ‘portable’ power models for DPA attacks.

ML key recovery with fully profiled templates

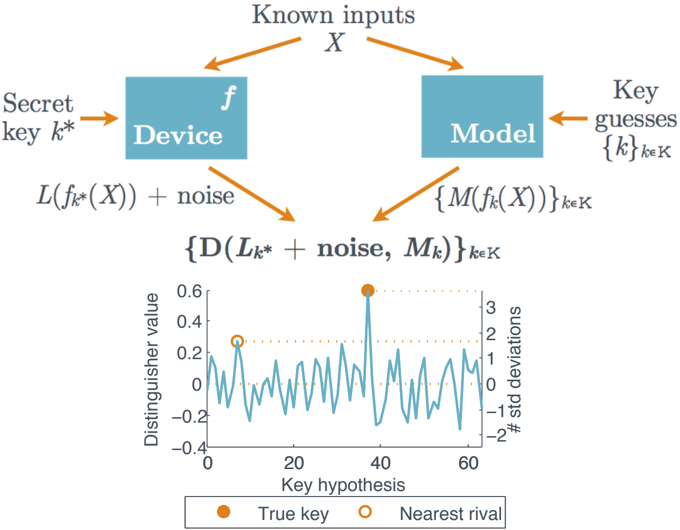
‘Standard’ DPA with ‘standard’ models (e.g. HW)

‘Standard’ DPA with approximated leakage models

Outline:

- Preliminaries: ‘Standard’ DPA; different ‘types’ of power model; unsupervised (k-means) clustering.
- Proposed methodology: unsupervised clustering for building nominal power models.
- Experimental results.

‘STANDARD DPA ATTACK’



Different types of power model

The power model M can approximate the deterministic part of the leakage L at different 'levels' . . .

Level	Correspondence	Associated attacks
Direct	$M \approx L$	Bayesian templates, stochastic profiling
Proportional	$M \approx \alpha L$	Pearson's correlation coefficient
Ordinal	$\{z \mid M(z) < M(z')\} \approx \{z \mid L(z) < L(z')\}\ \forall z' \in \mathcal{Z}$	Spearman's rank correlation coefficient
Nominal	$\{z \mid M(z) = M(z')\} \approx \{z \mid L(z) = L(z')\}\ \forall z' \in \mathcal{Z}$	'Partition'-based: mutual information, variance ratio, etc.

Different types of power model

The power model M can approximate the deterministic part of the leakage L at different 'levels' …

Level	Correspondence	Associated attacks
Direct	$M \approx L$	Bayesian templates, stochastic profiling
Proportional	$M \approx \alpha L$	Pearson's correlation coefficient
Ordinal	$\{z \mid M(z) < M(z')\} \approx \{z \mid L(z) < L(z')\}\ \forall z' \in \mathcal{Z}$	Spearman's rank correlation coefficient
Nominal	$\{z \mid M(z) = M(z')\} \approx \{z \mid L(z) = L(z')\}\ \forall z' \in \mathcal{Z}$	'Partition'-based: mutual information, variance ratio, etc.

Different types of power model

The power model M can approximate the deterministic part of the leakage L at different 'levels' ...

Level	Correspondence	Associated attacks
Direct	$M \approx L$	Bayesian templates, stochastic profiling
Proportional	$M \approx \alpha L$	Pearson's correlation coefficient
Ordinal	$\{z \mid M(z) < M(z')\} \approx \{z \mid L(z) < L(z')\}\ \forall z' \in \mathcal{Z}$	Spearman's rank correlation coefficient
Nominal	$\{z \mid M(z) = M(z')\} \approx \{z \mid L(z) = L(z')\}\ \forall z' \in \mathcal{Z}$	'Partition'-based: mutual information, variance ratio, etc.

Different types of power model

The power model M can approximate the deterministic part of the leakage L at different 'levels' ...

Level	Correspondence	Associated attacks
Direct	$M \approx L$	Bayesian templates, stochastic profiling
Proportional	$M \approx \alpha L$	Pearson's correlation coefficient
Ordinal	$\{z \mid M(z) < M(z')\} \approx \{z \mid L(z) < L(z')\}\ \forall z' \in \mathcal{Z}$	Spearman's rank correlation coefficient
Nominal	$\{z \mid M(z) = M(z')\} \approx \{z \mid L(z) = L(z')\}\ \forall z' \in \mathcal{Z}$	'Partition'-based: mutual information, variance ratio, etc.

Different types of power model

The power model M can approximate the deterministic part of the leakage L at different 'levels' ...

Level	Correspondence	Associated attacks
Direct	$M \approx L$	Bayesian templates, stochastic profiling
Proportional	$M \approx \alpha L$	Pearson's correlation coefficient
Ordinal	$\{z \mid M(z) < M(z')\} \approx \{z \mid L(z) < L(z')\}\ \forall z' \in \mathcal{Z}$	Spearman's rank correlation coefficient
Nominal	$\{z \mid M(z) = M(z')\} \approx \{z \mid L(z) = L(z')\}\ \forall z' \in \mathcal{Z}$	'Partition'-based: mutual information, variance ratio, etc.

Task: Arrange objects s.t. those inside a given group are *similar* whilst those in different groups are *dissimilar*.

Assumption: Number or characteristics of the underlying classes are *a priori* unknown (unlike supervised classification).

Method: Large selection of iterative trial-and-error solutions:

- Cluster models vary: hierarchical, centroid-based, density- or distribution-based, graph-based …
- 'Similarity' measures vary: Euclidean distance, correlation, Hamming, Manhattan …

N.B.: Notoriously difficult to match the best-suited learning algorithm to a given problem.

Unsupervised clustering

Task: Arrange objects s.t. those inside a given group are *similar* whilst those in different groups are *dissimilar*.

Assumption: Number or characteristics of the underlying classes are *a priori* unknown (unlike supervised classification).

Method: Large selection of iterative trial-and-error solutions:

- Cluster models vary: hierarchical, centroid-based, density- or distribution-based, graph-based …
- 'Similarity' measures vary: Euclidean distance, correlation, Hamming, Manhattan …

N.B.: Notoriously difficult to match the best-suited learning algorithm to a given problem.

Task: Arrange objects s.t. those inside a given group are *similar* whilst those in different groups are *dissimilar*.

Assumption: Number or characteristics of the underlying classes are *a priori* unknown (unlike supervised classification).

Method: Large selection of iterative trial-and-error solutions:

- Cluster models vary: hierarchical, centroid-based, density- or distribution-based, graph-based …
- 'Similarity' measures vary: Euclidean distance, correlation, Hamming, Manhattan …

N.B.: Notoriously difficult to match the best-suited learning algorithm to a given problem.

Unsupervised clustering

Task: Arrange objects s.t. those inside a given group are *similar* whilst those in different groups are *dissimilar*.

Assumption: Number or characteristics of the underlying classes are *a priori* unknown (unlike supervised classification).

Method: Large selection of iterative trial-and-error solutions:

- Cluster models vary: hierarchical, centroid-based, density- or distribution-based, graph-based …
- 'Similarity' measures vary: Euclidean distance, correlation, Hamming, Manhattan …

N.B.: Notoriously difficult to match the best-suited learning algorithm to a given problem.

Proposed methodology

General strategy

1. Partition the profiling traces according to the intermediate values and compute the means $\{\bar{\mathbf{t}}_z\}_{z \in \mathcal{Z}}$.
2. Obtain a mapping $M : \mathcal{Z} \longrightarrow \mathcal{M}$ by clustering the mean traces.
   - Values in $\mathcal{Z}$ not represented in the profiling dataset are mapped to cluster $C + 1$ (i.e. an 'other' category).
3. Use M as the (nominal) power model in 'partition-based' DPA against the target traces.

Example instantiation

Clustering algorithm: Principal component analysis followed by k-means clustering.

DPA distinguisher: Univariate and multivariate variance ratio.

Benchmark: Correlation DPA using the first principal component to approximate a 'proportional' power model.

Proposed methodology

General strategy

1. Partition the profiling traces according to the intermediate values and compute the means $\{\bar{\mathbf{t}}_z\}_{z\in\mathcal{Z}}$.
2. Obtain a mapping $M : \mathcal{Z} \longrightarrow \mathcal{M}$ by clustering the mean traces.
   - Values in $\mathcal{Z}$ not represented in the profiling dataset are mapped to cluster $C+1$ (i.e. an 'other' category).
3. Use M as the (nominal) power model in 'partition-based' DPA against the target traces.

Example instantiation

Clustering algorithm: Principal component analysis followed by k-means clustering.

DPA distinguisher: Univariate and multivariate variance ratio.

Benchmark: Correlation DPA using the first principal component to approximate a 'proportional' power model.

Proposed methodology

General strategy

1. Partition the profiling traces according to the intermediate values and compute the means $\{\bar{\mathbf{t}}_z\}_{z \in \mathcal{Z}}$.
2. Obtain a mapping $M : \mathcal{Z} \longrightarrow \mathcal{M}$ by clustering the mean traces.
   - Values in $\mathcal{Z}$ not represented in the profiling dataset are mapped to cluster $C + 1$ (i.e. an 'other' category).
3. Use M as the (nominal) power model in 'partition-based' DPA against the target traces.

Example instantiation

Clustering algorithm: Principal component analysis followed by k-means clustering.

DPA distinguisher: Univariate and multivariate variance ratio.

Benchmark: Correlation DPA using the first principal component to approximate a 'proportional' power model.

Proposed methodology

General strategy

1. Partition the profiling traces according to the intermediate values and compute the means $\{\bar{\mathbf{t}}_z\}_{z \in \mathcal{Z}}$.
2. Obtain a mapping $M : \mathcal{Z} \longrightarrow \mathcal{M}$ by clustering the mean traces.
   - Values in $\mathcal{Z}$ not represented in the profiling dataset are mapped to cluster $C + 1$ (i.e. an 'other' category).
3. Use M as the (nominal) power model in 'partition-based' DPA against the target traces.

Example instantiation

Clustering algorithm: Principal component analysis followed by *k*-means clustering.

DPA distinguisher: Univariate and multivariate variance ratio.

Benchmark: Correlation DPA using the first principal component to approximate a 'proportional' power model.

Proposed methodology

General strategy

1. Partition the profiling traces according to the intermediate values and compute the means $\{\bar{\mathbf{t}}_z\}_{z \in \mathcal{Z}}$.
2. Obtain a mapping $M : \mathcal{Z} \longrightarrow \mathcal{M}$ by clustering the mean traces.
   - Values in $\mathcal{Z}$ not represented in the profiling dataset are mapped to cluster $C + 1$ (i.e. an 'other' category).
3. Use M as the (nominal) power model in 'partition-based' DPA against the target traces.

Example instantiation

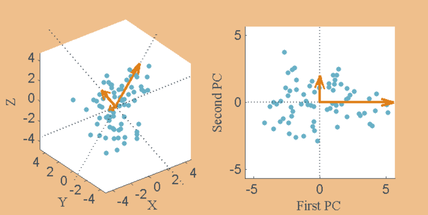
Clustering algorithm: Principal component analysis followed by k-means clustering.

DPA distinguisher: Univariate and multivariate variance ratio.

Benchmark: Correlation DPA using the first principal component to approximate a 'proportional' power model.

Principal Component Analysis

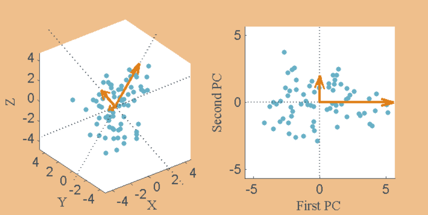
Transforms a large number of correlated variables into uncorrelated components (eigenvectors of covariance matrix). These are sorted in descending order of variance (eigenvalues of covariance matrix).



- Existing applications to side-channel analysis:
  - Preliminary step to Gaussian template building (avoids inversion problems caused by collinear 'points of interest').
  - Pre-processing to increase non-profiled DPA efficiency.
- Frequently used in unsupervised clustering to mitigate for sparseness (product space so large that *no* observations are 'close').
- Natural role in our clustering procedure: PCA on the mean traces finds the directions along which data-dependent variation is largest.

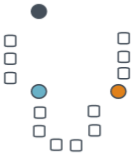
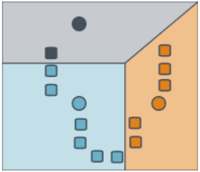
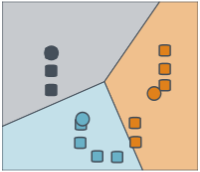
Principal Component Analysis

Transforms a large number of correlated variables into uncorrelated components (eigenvectors of covariance matrix). These are sorted in descending order of variance (eigenvalues of covariance matrix).



- Existing applications to side-channel analysis:
  - Preliminary step to Gaussian template building (avoids inversion problems caused by collinear 'points of interest').
  - Pre-processing to increase non-profiled DPA efficiency.
- Frequently used in unsupervised clustering to mitigate for sparseness (product space so large that *no* observations are 'close').
- Natural role in our clustering procedure: PCA on the mean traces finds the directions along which data-dependent variation is largest.

K-means clustering

Step 1	Step 2	Step 3	Step 4
			
Generate k initial “means” within the data domain.	Associate every observation with the nearest mean.	Compute the new means from the resulting clusters.	Repeat 2. and 3. until convergence is reached.

Problem: Quality of clustering depends on user-specified factors; 'best' choices *a priori* unknown.

- Optimal number of principal components to keep?
- 'Correct' number of clusters?

Silhouette index for i^{th} object…

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- a_i: mean distance from i^{th} object to other objects in its cluster;
- b_i: mean distance from i^{th} object to objects in nearest other cluster.

Strategy: Trial different combinations of settings and choose the one which produces the highest average silhouette index.

Problem: Quality of clustering depends on user-specified factors; 'best' choices *a priori* unknown.

- Optimal number of principal components to keep?
- 'Correct' number of clusters?

Silhouette index for i^{th} object…

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- a_i: mean distance from i^{th} object to other objects in its cluster;
- b_i: mean distance from i^{th} object to objects in nearest other cluster.

Strategy: Trial different combinations of settings and choose the one which produces the highest average silhouette index.

Choosing the best configuration

Problem: Quality of clustering depends on user-specified factors; 'best' choices *a priori* unknown.

- Optimal number of principal components to keep?
- 'Correct' number of clusters?

Silhouette index for i^{th} object…

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- a_i: mean distance from i^{th} object to other objects in its cluster;
- b_i: mean distance from i^{th} object to objects in nearest other cluster.

Strategy: Trial different combinations of settings and choose the one which produces the highest average silhouette index.

The variance ratio

$$D_{\mathrm{VR}}(k) = \frac{\sum_{t \in \boldsymbol{\tau}'} \mathrm{var}(\{P_{t,i}\}_{i=1}^{N})^2}{\frac{1}{N} \sum_{m \in \mathcal{M}} n_m \sum_{t \in \boldsymbol{\tau}'} \mathrm{var}(\{P_{t,i} | M \circ F_k(x_i) = m\})^2}$$

- $\boldsymbol{\tau}'$: attacker's best knowledge about $\boldsymbol{\tau}$ (want $\boldsymbol{\tau}' \cap \boldsymbol{\tau} \neq \emptyset$);
- M: nominal approximation (values in $\mathcal{M}$) for the leakage;
- $n_m = \#\{x_i | M \circ F_k(x_i) = m\}$, i.e. the number of observations in the trace set for which the predicted cluster label is m.

[See L. Batina, B. Gierlichs, and K. Lemke-Rust, *Differential Cluster Analysis*, CHES 2009, vol.5747 of LNCS, pp.112–127, Springer]

Sample variance of global trace distribution at time point t

$$D_{\mathrm{VR}}(k) = \frac{\sum_{t \in \boldsymbol{\tau}'} \mathrm{var}(\{P_{t,i}\}_{i=1}^{N})^2}{\frac{1}{N} \sum_{m \in \mathcal{M}} n_m \sum_{t \in \boldsymbol{\tau}'} \mathrm{var}(\{P_{t,i} | M \circ F_k(x_i) = m\})^2}$$

Sample variance of conditional trace distribution associated with a given model prediction

- $\boldsymbol{\tau}'$: attacker's best knowledge about $\boldsymbol{\tau}$ (want $\boldsymbol{\tau}' \cap \boldsymbol{\tau} \neq \emptyset$);
- M: nominal approximation (values in $\mathcal{M}$) for the leakage;
- $n_m = \#\{x_i | M \circ F_k(x_i) = m\}$, i.e. the number of observations in the trace set for which the predicted cluster label is m.

Experimental results

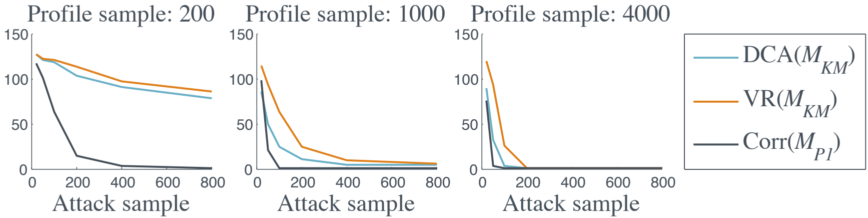
Data

Software: 10,000 traces from an unprotected AES implementation on an ARM microcontroller.

Hardware: 5,000 traces from an unprotected AES implementation on an RFID-type system.

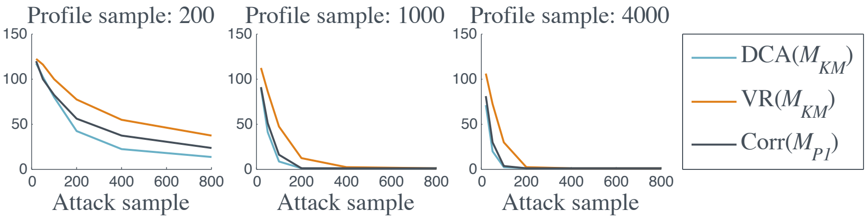
Experimental approach

1. Randomly draw (disjoint) profiling and attack samples from the full dataset.
2. Derive nominal and proportional power models from the profiling subsample.
3. Modify the attack subsample to simulate a variety of discrepancies.
4. Perform correlation- and univariate/multivariate VR-based DPA.
5. Repeat to estimate guessing entropies (average rank of correct subkey).



Guessing entropy of partially profiled DPA attacks against an unprotected software implementation of AES. Window width: 20; reps: 500.

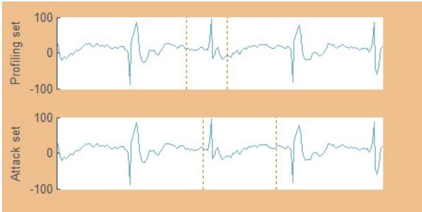
- Clustering strategy ‘works’: uncertainty about the subkey is reduced.
- Multivariate distinguisher outperforms the univariate one.
- Correlation DPA with our estimated proportional model is more efficient in terms of number of attack *and* number of profiling traces needed.



Guessing entropy of partially profiled DPA attacks against an unprotected hardware implementation of AES. Window width: 10; reps: 500.

- Implementation: two 32-bit registers; byte substitutions occur in parallel with MixColumns operation in previous column.
- Considerable variation in the exploitability of the S-boxes (we report for the most vulnerable one).
- Multivariate distinguisher now outperforms correlation DPA.

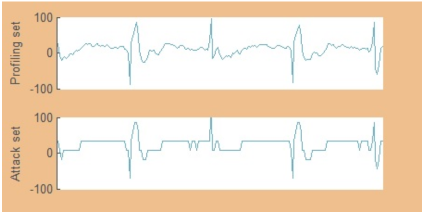
Discrepancy in window width and location



Scenario: Attacker roughly knows the interesting 'windows' but cannot match them precisely.
Simulated distortion: Pick different window sizes and offsets in the attack subsample.

Attack sample size ⟶	Software						Hardware					
	DCA(M_{KM})		**VR**(M_{KM})		**Corr**(M_{P1})		**DCA**(M_{KM})		**VR**(M_{KM})		**Corr**(M_{P1})	
	50	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**
Offset $-\lfloor \mathbf{w}/2 \rfloor$	53	1	87	1	15	1	121	65	68	1	22	1
$-\lfloor \mathbf{w}/4 \rfloor$	37	1	65	1	3	1	51	1	66	1	20	1
0	34	1	72	1	1	1	15	1	65	1	21	1
$\lfloor \mathbf{w}/4 \rfloor$	27	1	83	1	1	1	25	1	76	1	24	1
$\lfloor \mathbf{w}/2 \rfloor$	74	4	109	1	22	1	66	1	113	3	90	1

- Software attacks vulnerable to this; larger samples help to compensate.
- Hardware attacks vulnerable to the most extreme shifts.

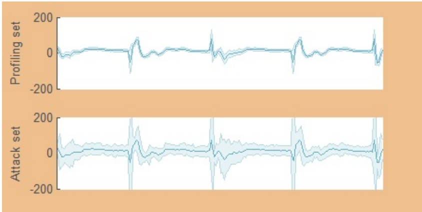


Scenario: Training and target traces are collected at different resolutions (e.g. due to different equipment). Simulated distortion: Discretise the attack subsample into fewer numbers of equally-sized bins.

Attack sample size ⟶		Software						Hardware					
		DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})		DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})	
		50	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**
Number of bins	**256**	30	1	86	1	5	1	16	1	68	1	23	1
	128	28	1	83	1	5	1	16	1	66	1	21	1
	64	38	1	81	1	9	1	17	1	62	1	29	1
	32	68	1	107	1	29	1	20	1	65	1	32	1
	16	70	1	135	133	26	1	33	1	71	1	55	1

- Some evidence of eventual decline in attack effectiveness as measurements reach their most granular.

Discrepancy in measurement error

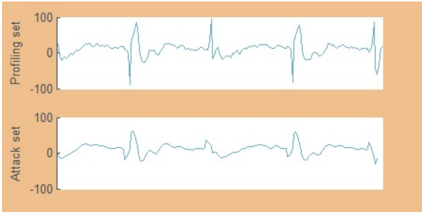


Scenario: Target traces are noisier than training traces (e.g. due to inferior measurement set-up).
Simulated distortion: Add a (zero mean) Gaussian-distributed random sample to each measurement.

Attack sample size ⟶	Software						Hardware					
	DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})		DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})	
	50	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**
Noise factor **1**	31	1	93	1	9	1	22	1	86	1	29	1
2	71	1	103	1	33	1	56	1	107	1	65	1
4	100	3	118	8	78	1	71	1	100	14	80	2
8	124	14	115	38	103	1	116	7	123	50	95	9
16	115	52	133	107	129	14	112	40	113	85	114	67

- As expected: all three attacks remain effective, but the number of traces required for equivalent success scales proportionally.

Discrepancy in trace pre-processing

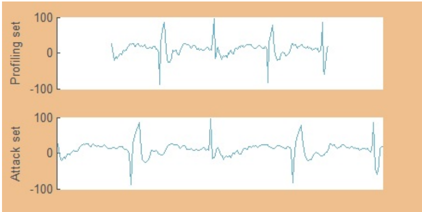


Scenario: Training traces have been pre-processed in a manner not precisely known to the attacker.

Simulated distortion: Apply additional filtering to the attack subsample (moving averages).

Attack sample size ⟶	**Software**						**Hardware**					
	DCA(M_{KM})		**VR(M_{KM})**		**Corr(M_{P1})**		**DCA(M_{KM})**		**VR(M_{KM})**		**Corr(M_{P1})**	
	50	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**	**50**	**400**
Smoothing window 1	43	1	96	1	16	1	19	1	62	1	19	1
2	44	1	75	1	5	1	24	1	59	1	17	1
4	51	1	104	1	5	1	74	1	100	4	79	1
8	77	1	106	1	16	1	111	32	121	54	100	17
16	115	5	123	3	53	1	112	82	118	94	113	64

- Software attacks robust; smoothing pairwise even improves outcomes.
- Hardware attacks less robust (fewer clock cycles; raw traces are already shorter and more coarsely sampled).



Scenario: Misalignment caused by varying frequency in target traces (e.g. for ‘hiding’).

Simulated distortion: ‘Pad’ a proportion of sample points with additional values in random positions.

Attack sample size ⟶	Software						Hardware					
	DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})		DCA(M_{KM})		VR(M_{KM})		Corr(M_{P1})	
	50	400	50	400	50	400	50	400	50	400	50	400
Insertions (prop.) 0.005	133	125	131	124	139	137	122	125	122	97	117	46
0.01	126	111	134	119	128	135	135	127	123	146	139	108
0.05	120	135	133	123	131	123	125	117	126	127	125	131
0.1	141	134	131	127	129	134	131	116	138	135	126	135
0.5	130	113	138	121	116	131	143	131	128	138	134	131

- All attacks fail; correct key ranking does not improve, even as number of traces increases.

In conclusion...

- Unsupervised clustering can recover nominal power models for use in effective ‘partition-based’ DPA.
  - Requirements in profiling phase are minimal relative to full profiling.
  - Robustness to discrepancies between profiling and attack traces is considerably greater.
- Proportional power models can recovered under the same circumstances, for use in correlation DPA.
  - More efficient, in the case of software experiments; slightly less in the case of hardware experiments.
  - Almost as robust.
- Open question: Are there clustering algorithms which perform better?

Thank you for listening! Any questions?