NPTEL ONLINE CERTIFICATION COURSES

Course Name: Hardware Security

Faculty Name: Prof Debdeep Mukhopadhyay

Department : Computer Science and Engineering

Topic

Lecture 32: Power Analysis-VII

CONCEPTS COVERED

Concepts Covered:

- Metrics to Evaluate Side Channel Analysis
- CPA on Real Traces of AES-128
- SNR of Power Traces
- DPA of Iterative Architectures, Serial Architectures and Shuffled Schemes

Metrics for SCA

- In order to evaluate the attacks and also to compare the crypto-implementations wrt. the SCAs we need quantifiable metrics.
- Useful Metrics:
  - Success Rate of a SCA adversary
  - Guessing Entropy of an Adversary

Success Rate of an Adversary

- SCA works as a divide and conquer strategy, where the key space is divided into several equivalent classes.
- The attack normally does not distinguish between keys which belong to the same class or partition.
- We can formalize the cryptographic implementation as E_K, where K is the key space.
- The adversary assumes a leakage model, denoted by L
  - The leakage model provides some information about the key or some other desirable information.
- The adversary is an efficient algorithm denoted by $A_{E_K,L}$: bounded by time complexity, τ, memory complexity, m, and q queries.

Formal Definition of Success Rate

- The leakage exploited by the adversary , maps a key $k \in K$, to a set, within which it cannot distinguish.
- We define these partitions by S, and the mapping by a function γ, st. $s = \gamma(k), k \in K$.
- Typically, $|S| \ll |K|$.
- The objective of the attack is to determine the class s to which a target k belongs with non-negligible probability.
- As an analogy, consider the Hamming Weight class, which divides the key space into equivalence classes or partitions.

Formal Definition of Success Rate

- As an analogy, consider the Hamming Weight class, which divides the key space into equivalence classes or partitions.
- The output of the adversary is based on the ciphertexts (black-box information) and the leakage (side channel information).
- The output is a guess-vector, which are the key classes sorted in terms of descending order of being a likely candidate.
- Thus we can define a order-o ($o \leq |S|$) adversary when the adversary produces the guessing-vector as $g = [g_1, g_2, \cdots, g_o]$

Formal Definition of Success Rate

Input: K, L, E_K
Output: 0 (failure), 1 (success)

1 $k \in_R K$
2 $s = \gamma(k)$
3 $\mathbf{g} = [g_1, \ldots, g_o] \leftarrow A_{E_k, L}$
4 **if** $s \notin g$ **then**
5 **return** 0
6 **end**
7 **else**
8 **return** 1
9 **end**

The experiment is a success if $s \in g$

$$Succ_{A_{E_K,L}(\tau,m,q)} = \Pr[Exp_{A_{E_K,L}} = 1]$$

ie. If the correct key class belongs in the guessing vector the attack is successful.

Guessing Entropy of an Adversary

- The above metric for an o^{th}order attack implies the success rate for an attack where the remaining load is o-key classes.
- The attacker has a maximum of o-key classes to which the required k may belong.
- While the above definition is fixed wrt. the remaining work load, we define guessing-entropy to provide a more flexible definition for the remaining work load.

Formalizing Guessing Entropy

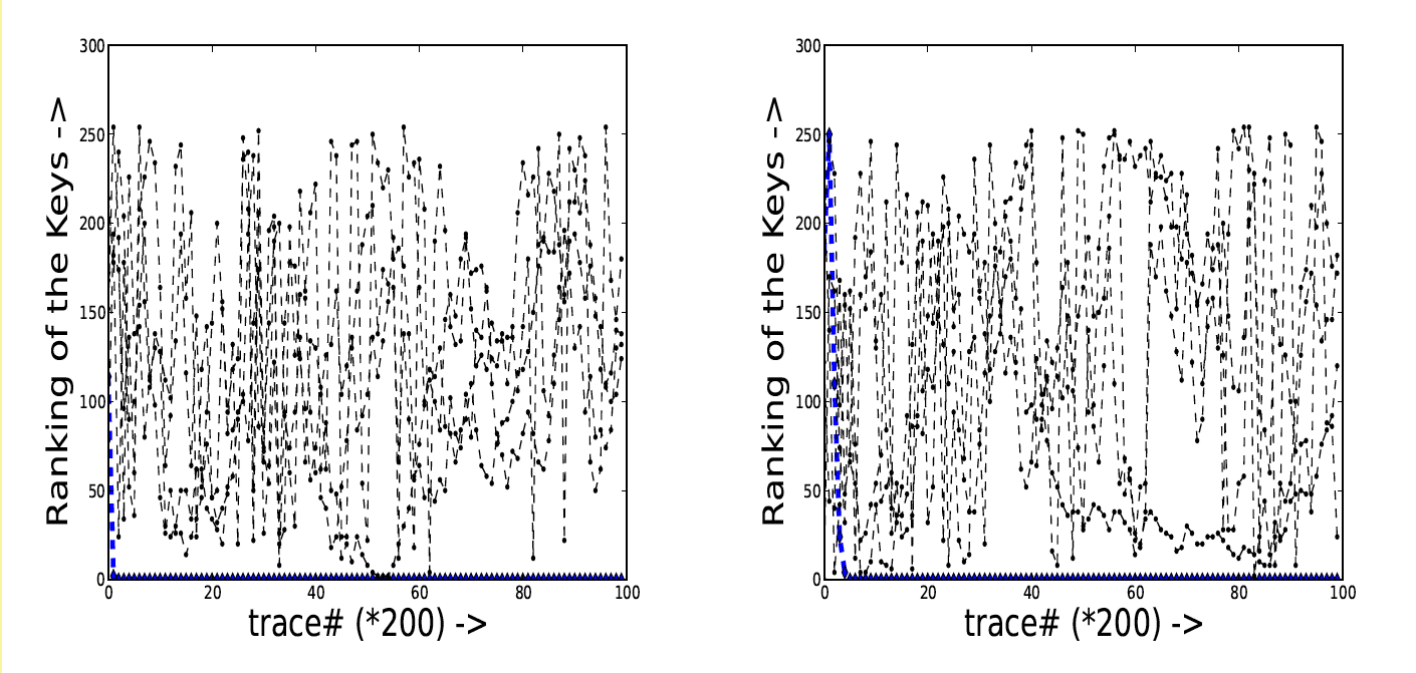
Input: K, L, E_K
Output: Key class i

1 $k \in_R K$
2 $s = \gamma(k)$
3 $\mathbf{g} = [g_1, \ldots, g_o] \leftarrow A_{E_k, L}$
4 **return** i such that $g_i = s$

$$GE_{A_{E_K,L}(\tau,m,q)} = E[Exp_{A_{E_K,L}}]$$

- Formal definition of Guessing Entropy of the adversary against a key class variable S
- Experiment returns the position of the candidate key class in the guessing entropy vector.

Guessing Entropy Plots



CPA on Simulated Power Traces with and without noise.
Blue indicates that the guessing entropy for the correct key byte drops to 0 faster than the wrong keys.

CPA on Real Power Traces

- When attacking real power traces, the effect of electrical and algorithmic noise comes into effect.
- Architecture often has a strong effect on the algorithmic noise.
- It determines the Signal-to-Noise Ratio (SNR) which quantifies the quality of the power traces.
- This in turn affects the Success rate and the Guessing-Entropy of the attacks.

SNR of a Power Trace

- Consider a Cipher, like AES, with r-rounds and let S be a random variable representing the key dependent intermediate variable of E.
- S is called the target and satisfies $S = F_{k^*}(X)$, where X is a random variable representing a part of known plaintext or ciphertext.
- F_{k^*} is a function which depends on a part of the cipher and the output depends on a part of the secret key, denoted as k^*.
- The function F_{k^*} also depends on the leakage function of the hardware device.
- Thus the total leakage at time instance t, denoted by the rv. L_t, can we written as:

$$\boldsymbol{L_t = aF_{k^*}(X) + N_t = aS + N_t}$$

Here a is some real constant. N_t is some Gaussian Distribution $N_t \sim N(0, \sigma_t)$

Univariate vs Multivariate Distinguishers

- In the above equation, aS is the deterministic part of the leakage.
  - It can be simulated by the adversary knowing the values of X and for each hypothesis of the keys.
- The time instance when the target S is manipulated is called as the instance of interest, say t^*.
- However, in a practical scenario t^* is unknown.
- In the most common form of DPA, an attacker applies a univariate distinguisher at each time instance, $0 \leq t < rT$, where r is the number of rounds, T is the number of samples per round.
- Thus if the observed leakage be denoted as $F_k(X)$, the attacker computes the distinguisher vector $D_t = (\{d_k\}_{k \in K})_t$, where d_k is a univariate distinguisher, denoted as: $\boldsymbol{d_k = D(aF_{k^*}(X) + N_t, F_k(X))}$

The final result is based on the best of all these time instances.

Univariate vs Multivariate Distinguishers

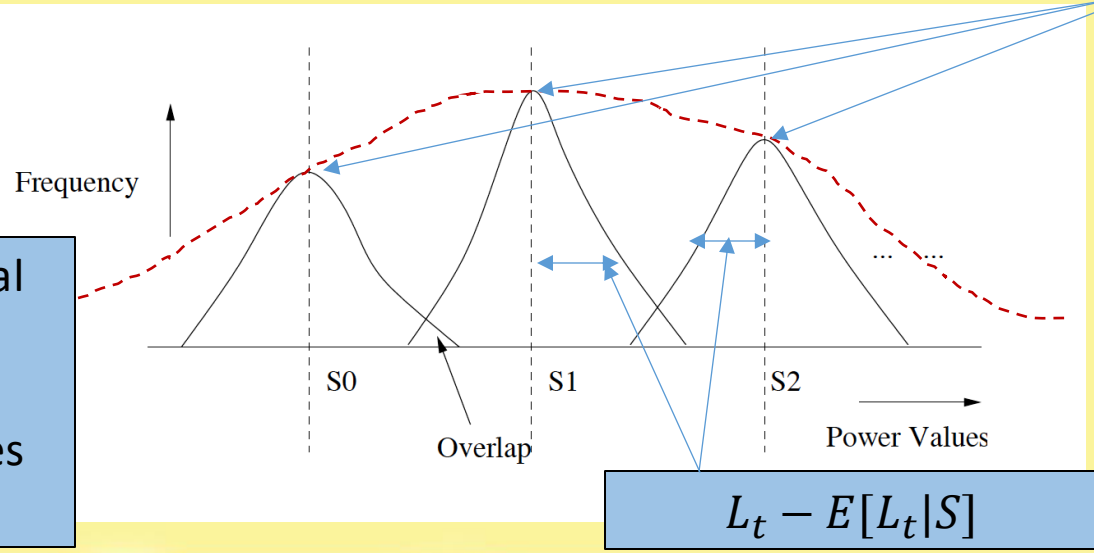
- In the multivariate approach, the attacker uses multivariate distinguisher which jointly evaluates the power consumptions at multiple sample points.
- Very common in profiling attacks like Template attacks.
- However vulnerable to a decrease in success rate due to some points in the trace having less signal component (leakage due to deterministic portion), compared to noise. This is often captured by SNR (Signal-to-Noise Ratio)
- Hence, when high SNR points in the trace are combined with low SNR points leads to the decrease in overall success rates.

Definition of SNR

- A register X is being updated by an AES circuit depending on the inputs.
  - We observe the power traces by varying the inputs and creating a frequency plot at some point of interest, t
  - Our power model tells that the power values would depend on the Hamming weights, say S.

$Var(E[L_t|S])$ indicates the variations in leakage due to the target S at sample point t.

This contributes to the signal component, in aiding the attack.
More this value, the S-values are distinguishable better



$E[L_t|S]$

$Var(L_t - E[L_t|S])$ contributes to noise.
More this variance, more is the overlap between the leakage due to the target variable S. This hinders the attack.

$L_t - E[L_t|S]$

Definition of SNR

- $SNR_t = \frac{Var(E[L_t|S])}{Var(L_t - E[L_t|S])}$
  - The SNR is a very useful metric for assessing the threat of a power attack on a given implementation.
- How can it be computed in experiments?
  - Say we collect 10,000 traces.
  - We target a byte, say denoted as S.
  - Depending on the Hamming Weight (say) we split the traces into 9 bins.
  - At a time t, thus the i^{th}-bin consists of power values, $P_i^0, \cdots, P_i^{n_i-1}$
    - The average of this gives a point in the distribution $E[L_t|S]$. We calculate the variance for the signal.
  - From each of the power values in the i^{th}-bin we subtract the average of the bin.
    - The variance of this gives the noise component.

Relating Correlation with SNR

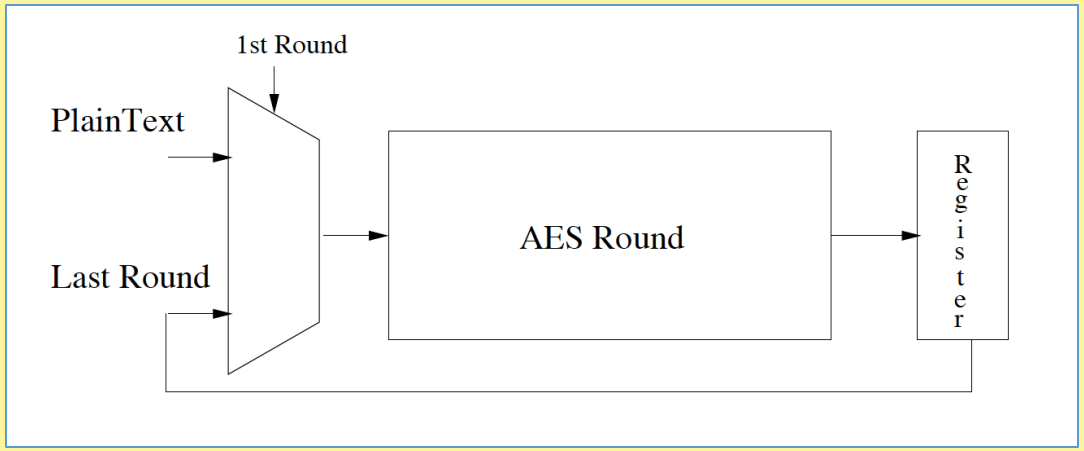
- $L_t = P_{det} + N$, where P_{det} is the deterministic component of power and N is the noise.
- Thus the correlation between the total leakage L_t and the hypothetical power value for the i^{th} key H_i as $Corr(H_i, L_t)$:
- $Corr(H_i, L_t) = Corr(H_i, P_{det} + N) = \frac{Cov(H_i,(P_{det}+N))}{\sqrt{Var(H_i)\left(Var(P_{det})+Var(N)\right)}}$
- Thus, $Corr(H_i, L_t) = \frac{E\left(H_i.(P_{det}+N)\right)-E(H_i)E(P_{det}+N)}{\sqrt{Var(H_i)Var(P_{det})}\sqrt{1+\frac{Var(N)}{Var(P_{det})}}}$

Relating Correlation with SNR

- If H_i and N are independent, $Cov(H_i, \mathrm{N}) = 0 \Rightarrow 0 = E(H_i.N) - E(H_i)E(N) \Rightarrow E(H_i.N) = E(H_i)E(N)$.
- $\therefore E\big(H_i.(P_{det} + N)\big) - E(H_i)E(P_{det} + N) = E(H_i.P_{det}) + E(H_i.N) - E(H_i)E(P_{det}) - E(H_i)E(N) = E(H_i.\ P_{det}) - E(H_i)E(P_{det})$
- Thus, $Corr(H_i, L_t) = \frac{E(H_i.\ P_{det}) - E(H_i)E(P_{det})}{\sqrt{Var(H_i)Var(P_{det})}\sqrt{1+\frac{Var(N)}{Var(P_{det})}}} = \frac{Corr(H_i.\ P_{det})}{\sqrt{1+\frac{1}{SNR}}}$

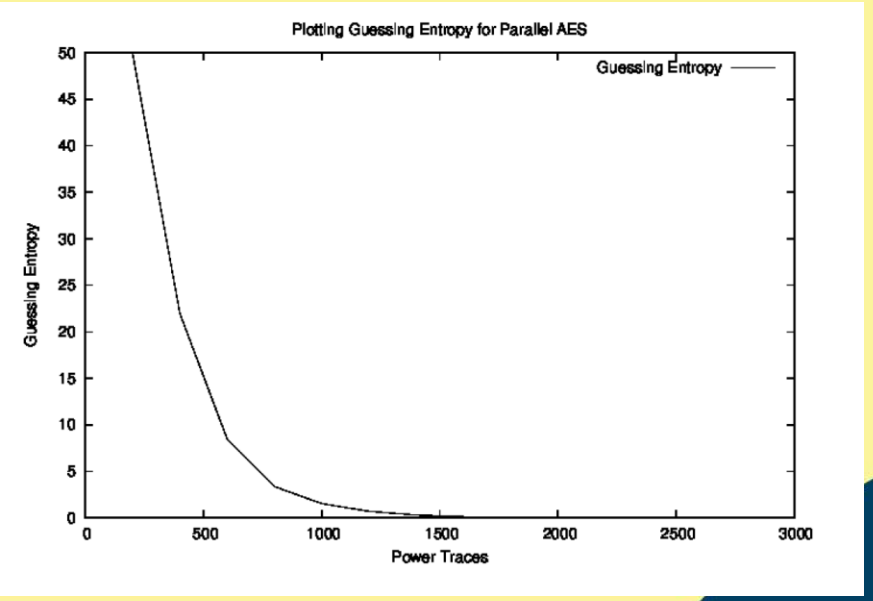
The above result can be used to evaluate an architecture based on simulated power traces. The simulation computes the value of $Corr(H_i.\ P_{det})$. However, the real correlation can be computed using the SNR.

DPA on an Iterative Architecture with Parallel S-Boxes

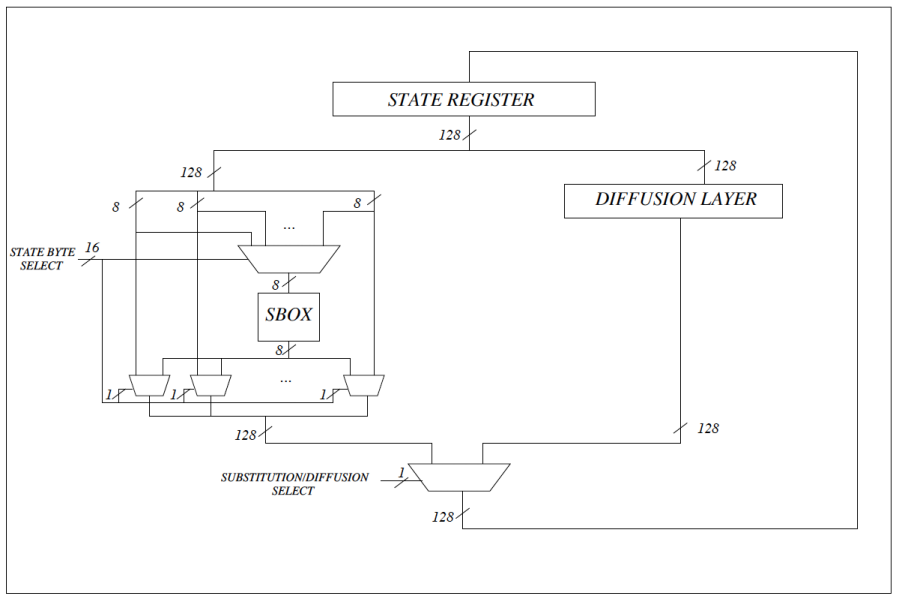


Power traces are collected and correlated with hypothetical power values to perform CPA.
All the S-Boxes are in parallel in this architecture, providing algorithmic noise.

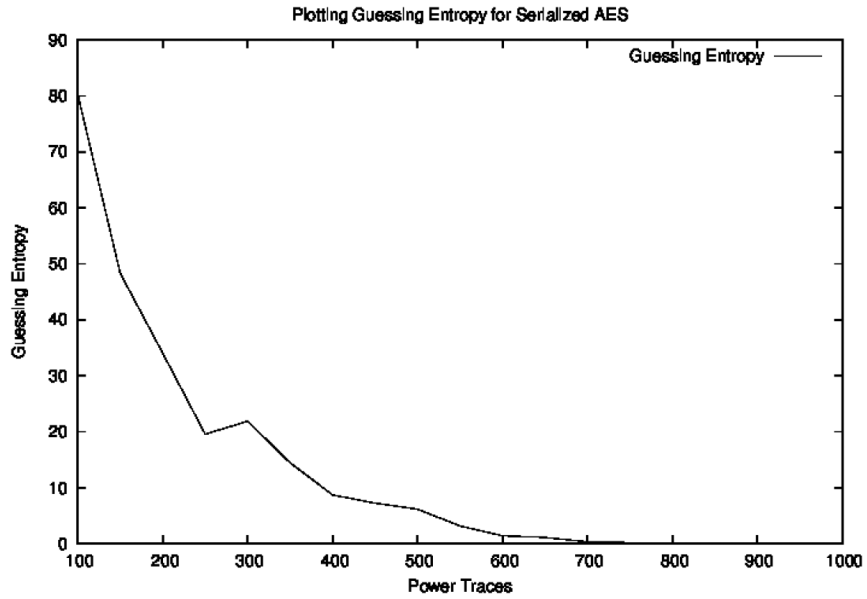
We acquired 70,000 power traces and divided into sets of 3000 traces each. We perform the CPA attacks on them, and plot the average Guessing Entropy for 3000 traces.



DPA on a Serialized Architecture of AES



- Serialized architecture, wherein there is only one S-Box.
- A MUX is used to select out one byte from the 128-bit state and process by the S-Box.
- Less algorithmic noise.



Plot of average Guessing Entropy for 1000 traces, averaged over 40 sets.
Compared to the parallel architecture, one can see the less number of observations required.

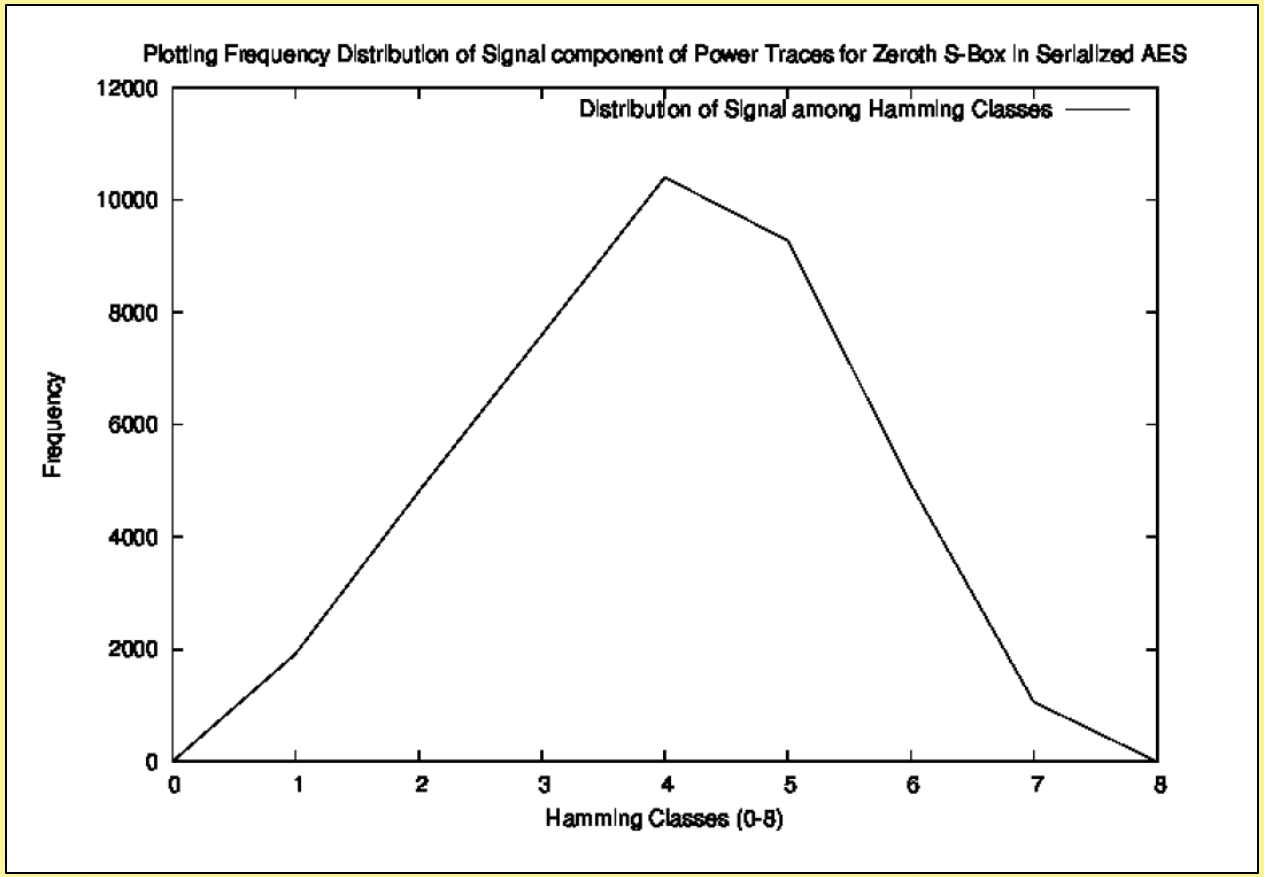
Shuffled Architecture and SNR

- A very popular technique to increase the resistance against CPA is called shuffling.
- In this architecture, the shuffling scheme randomizes the sequence of S-Box operations.
- Does not have a negative effect on throughput, as in when we defend using dummy operations.
- Quite easy to implement: make the select line in the multiplexer arbitrary.
- However, if one observes the total power in 16 clock cycles, one can still get the CPA working!

Comparing based on SNR Computations

- Serialized Architecture:
  - We observe the power consumption say during the 0^{th} S-Box computation.
  - We compute the Hamming Distance (or weight) corresponding to the 0^{th} S-Box. Note for this we need the key.
    - Classes can be from 0 to 8.
  - We calculate the average of each class: p_i, for $0 \leq i \leq 8$.
  - The number of elements in each class is say f_i.
  - Thus, Signal is $Var(E[L_t|S]) = \frac{\Sigma_{i=0}^{8} f_i p_i^2}{\Sigma_{i=0}^{8} f_i} - \left(\frac{\Sigma_{i=0}^{8} f_i p_i}{\Sigma_{i=0}^{8} f_i}\right)^2$
  - For the noise component from each power trace, deduct the average p_i if the trace belongs to the i^{th} class. Calculate the variance across all the classes.

Frequency Plot of Signal Component of Power Traces for the 0-th S-Box of Serialized Architecture

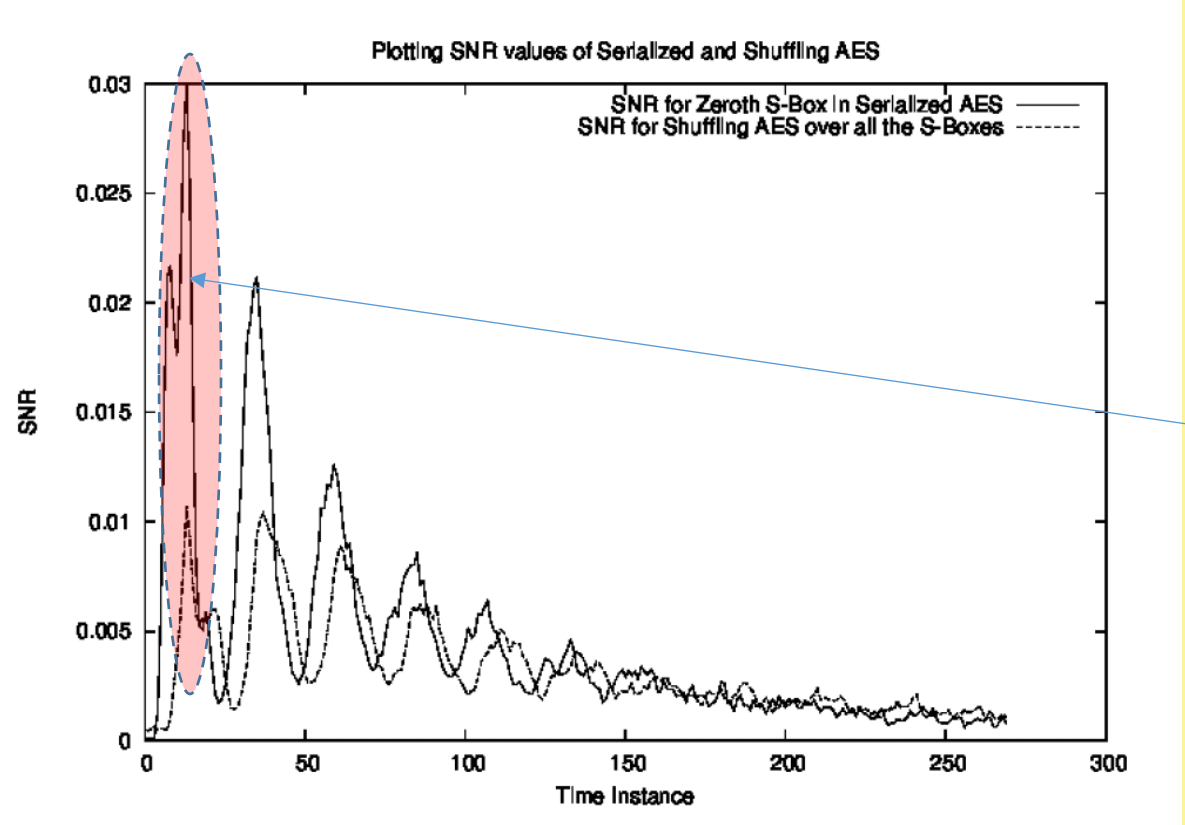


- The peak around 4 is expected.

SNR for Shuffled Architectures

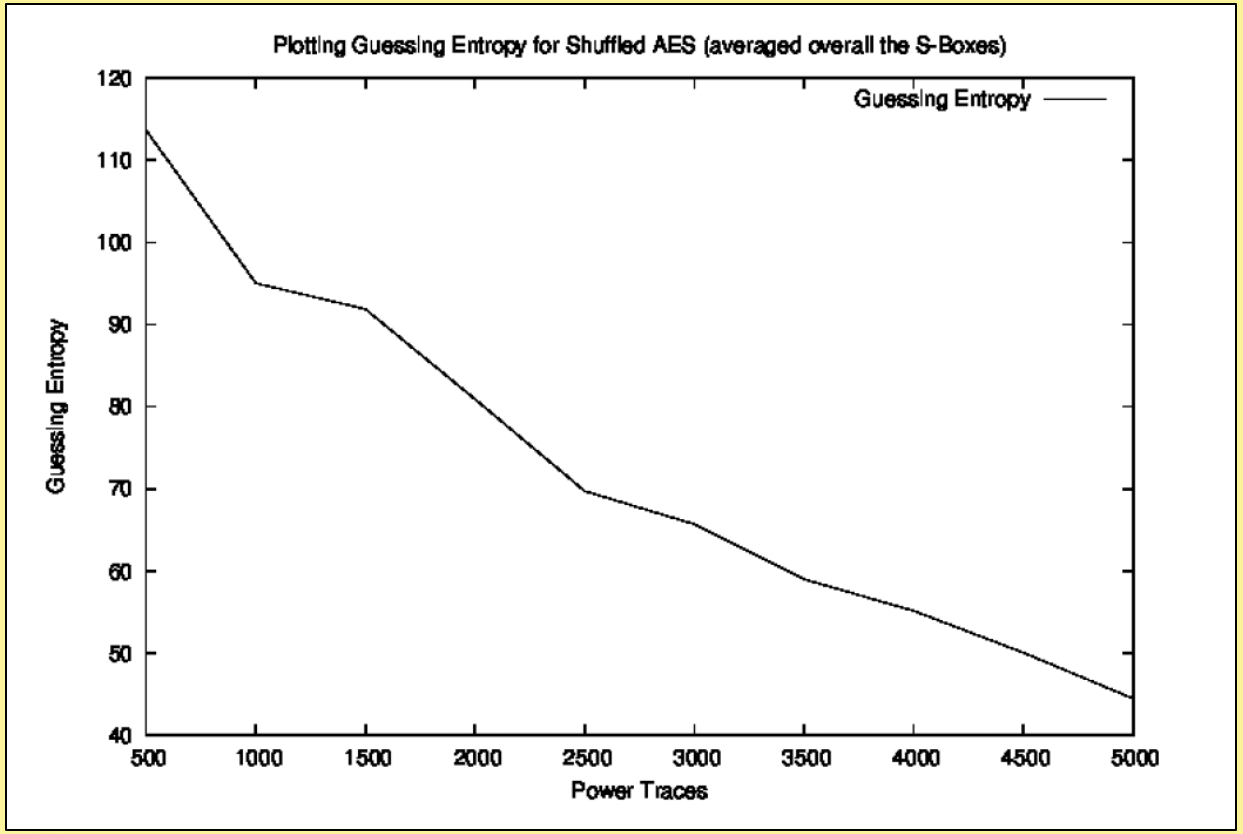
- For the shuffled scheme, we compute the SNR for the total power for all the 16 S-Boxes.
- This increases as if we target only one S-Box, then the SNR would be (1/256)-th that of the serialized implementation.
- We sum the power values in the traces for the sample points corresponding to each of the 16 cycles, and then they are divided into the Hamming Classes.
- It may be noted that since we are considering the total power for all the S-Boxes, the Hamming classes vary from 0 to 128.

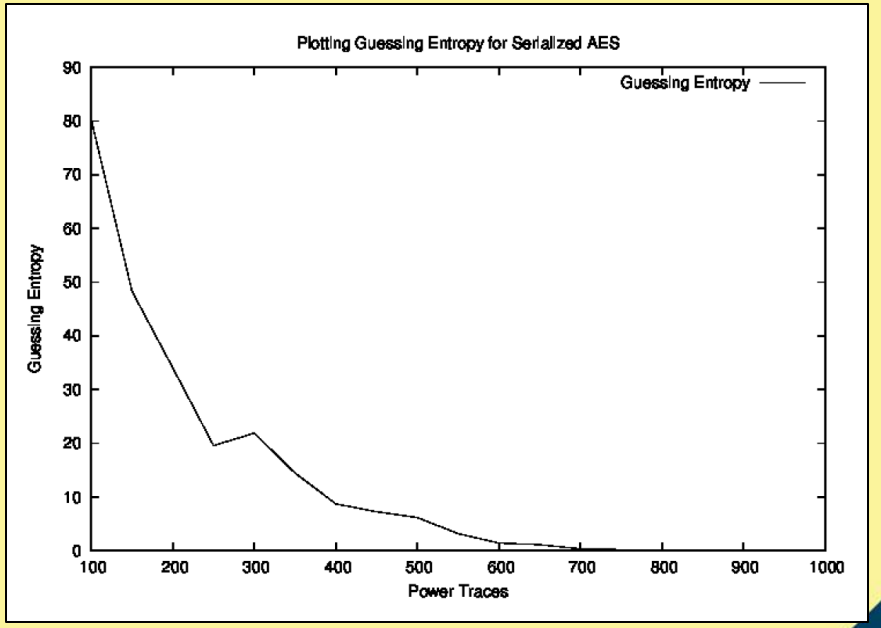
Comparison of SNRs



The average SNR for the shuffled scheme is around 1/3rd that of the serialized architecture.

Plot of Guessing Entropy for CPA on Shuffled vs Serialized Architectures



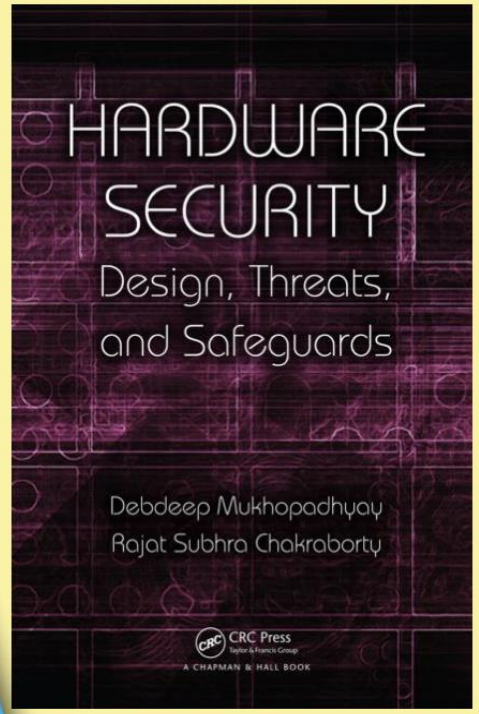


The GE reduces slowly compared to the serialized architecture

References

References:

- Debdeep Mukhopadhyay and Rajat Subhra Chakraborty, Hardware Security: Design, Threats and Safeguards, CRC Press



Conclusion:

Success Rate, Guessing Entropy (GE) are useful quantifiable metrics to evaluate SCA.

SNR of a power trace quantifies the quality of a power trace.

Correlation used in CPA is directly proportional to SNR.

DPA on Iterative Architecture, Serialized Architecture and Shuffled Architecture compared using SNR and GE.

FREE ONLINE EDUCATION
swayam
शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSES

Thank you!