Implementation Security In Cryptography

Lecture 11: SCA Countermeasures

Recap

- Till the last lecture
  - Side-Channel attacks, evaluation, metrics

Today

- Masking Against SCA

Countermeasure So Far

- We have briefly seen shuffling
- We have briefly experienced random delay
- Both have limited protection
  - Shuffling increases the number of traces linearly
  - Random delay can be undone with a some preprocessing (or DL).

Today’s Countermeasure

- Masking
  - Provable security against SCA
  - Exponential security amplification (with some noise)
  - Very well-established

Back to the Leakage Models

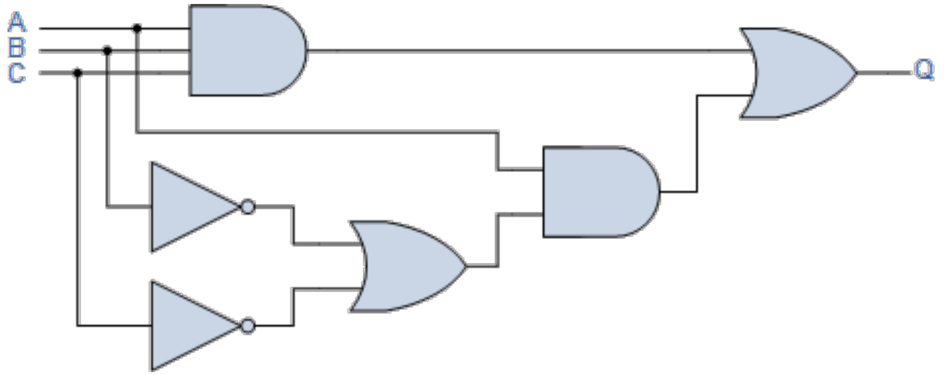
Simple Power Model. *Let t denote the time, and $\mathcal{N}(t)$ be a normal distributed random variable which represents the noise components. Let $f(g,t)$ denote the power consumption of gate g at the time t. Then a simplified power model for the power consumption is the function*

$$P(t) = \sum_{g} f(g,t) + \mathcal{N}(t)$$

- A leakage model leaks a function of the bits processed by a gate
  - So far, we tried to “learn” the leakage model as a function of several bits
  - Now, we go for a simpler yet detailed abstraction

Abstraction of Leakage: Noisy Leakage Model

- Leakage happens from every wire
- **The leakage is a noisy function of the value processed in every wire**
- SCA attack is basically the probability of the adversary distinguishing between two distribution of power traces:
    - Let the adversary be $\mathscr{A}$, and its interaction with a distribution D is $\mathscr{A}^D$. The distribution is basically the from traces — **noisy values of the wires**.
        - We are interested in *distinguishing probability* of $\mathscr{A}$



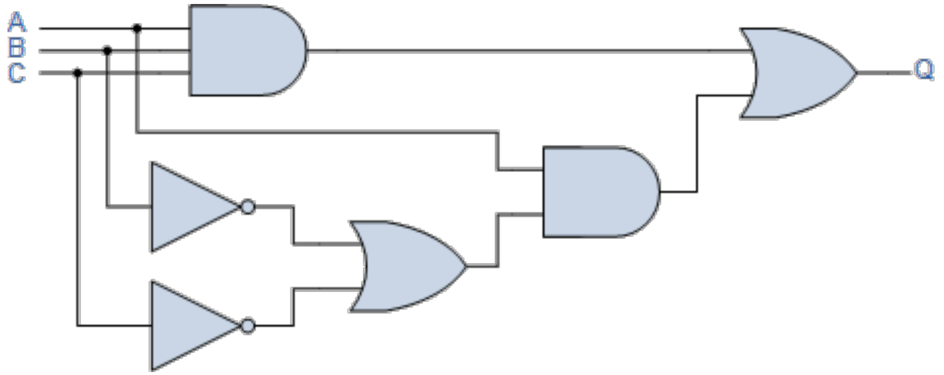
$$P(t) = \sum_g f(g,t) + \mathcal{N}(t)$$

Chari S, Jutla CS, Rao JR, Rohatgi P (1999) Towards sound approaches to counteract power-analysis attacks. In: CRYPTO. Lecture notes in computer science, vol 1666. Springer, pp 398–412

Abstraction of Leakage: In Simple Words

- Let the adversary be $\mathscr{A}$, and its interaction with a distribution D is $\mathscr{A}^D$.
  - We are interested in *distinguishing probability* of $\mathscr{A}$
  - Let the adversary gets power traces corresponding to two different distributions D_0 and D_1, corresponding to two different secret bits 0, 1.
  - The distinguishing probability is:

$$\left| Pr[A^{D_0} \to 1 - Pr[A^{D_1} \to 1] \right|$$

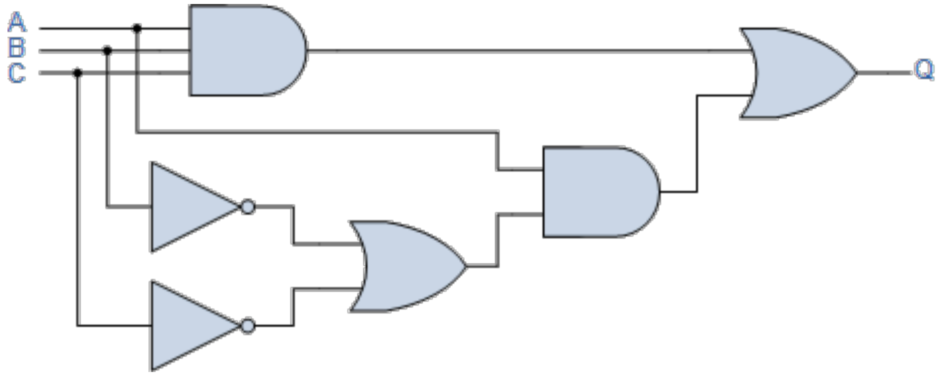


$$P(t) = \sum_g f(g,t) + \mathcal{N}(t)$$

Chari S, Jutla CS, Rao JR, Rohatgi P (1999) Towards sound approaches to counteract power-analysis attacks. In: CRYPTO. Lecture notes in computer science, vol 1666. Springer, pp 398–412

Abstraction of Leakage: Probing model

- Also called *threshold probing model*
  - Say you have a circuit with *n* wires
  - The adversary can probe t wires of its choice where $t < n$.
  - The adversary gets exact values of these wires
  - Now:
    - We shall be using probing model to argue our security.
    - But these two models are related !!! I will say a few words on them at the end…

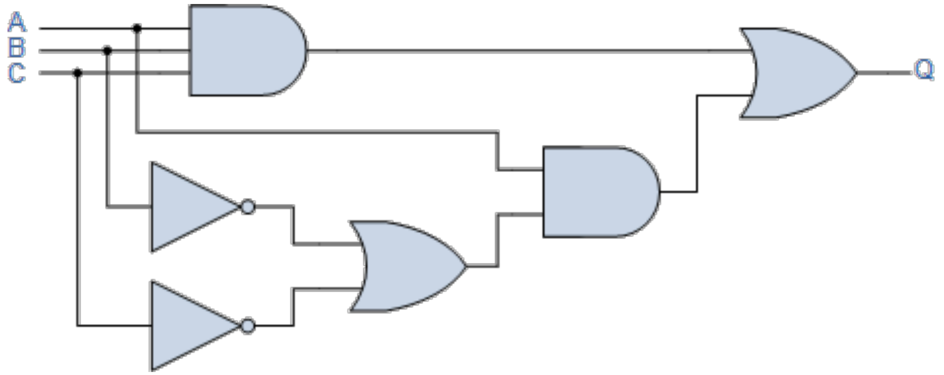


$$P(t) = \sum_g f(g,t) + \mathcal{N}(t)$$

Ishai Y, Sahai A, Wagner DA (2003) Private circuits: securing hardware against probing attacks. In: CRYPTO. Lecture notes in computer science, vol 2729. Springer, pp 463–481

What is Masking?

- Basically a way to make the power consumption uncorrelated with the internal computation
  - Randomize the computation/data
    - Even if you keep the secret same, every time the circuit operates on bits, all of which are random.
  - But maintain correctness
  - You have to ensure that none of the wires carry a bit which is correlated to the actual internal computation.
- So, it is totally an algorithmic trick/strategy.

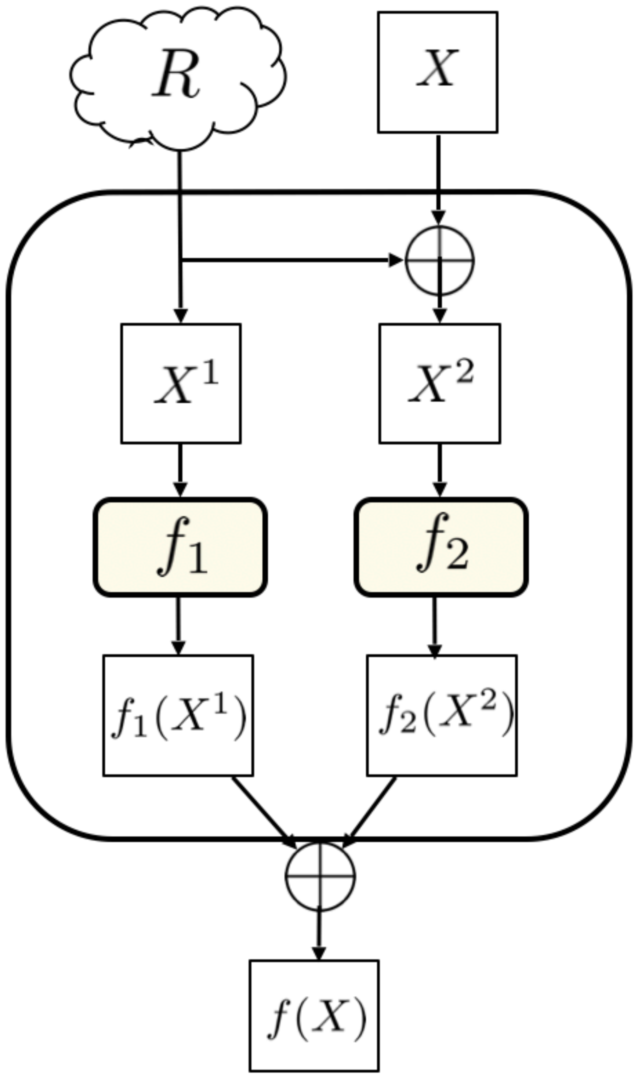


Ishai Y, Sahai A, Wagner DA (2003) Private circuits: securing hardware against probing attacks. In: CRYPTO. Lecture notes in computer science, vol 2729. Springer, pp 463–481

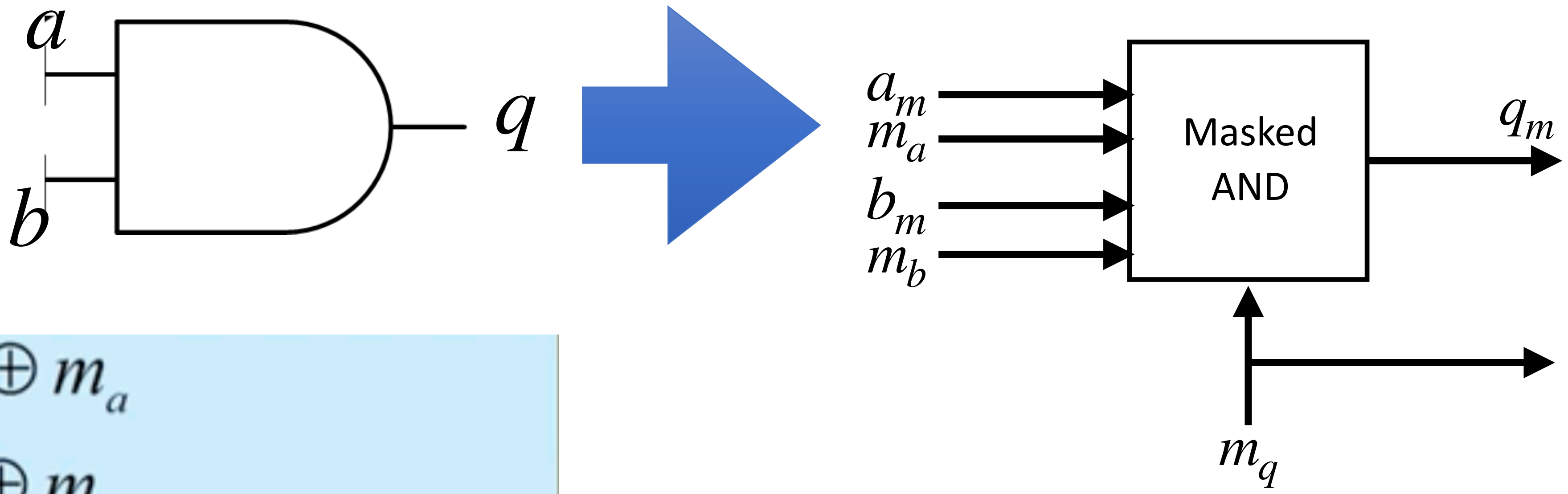
What is Masking?

Masking:

- SCA countermeasure.
- Makes power consumption independent of processed data
- Requires fresh randomness at every execution.
- Split a value X into multiple shares $\langle X^1, X^2, \cdots, X^n \rangle$ such that $X^1 + X^2 + \cdots + X^n = X$.
- Function $f(X)$ is split into functions $\langle f_1, f_2, \cdots, f_n \rangle$ such that $f_1(X^1) + f_2(X^2) + \cdots + f_n(X^n) = f(X)$.
- Splitting is trivial for linear functions.
- Nonlinear functions require special attention.



The First Example



$$a_m = a \oplus m_a$$

$$b_m = b \oplus m_b$$

$$q_m = q \oplus m_q$$

$$q = f(a, b)$$

$$q_m = \hat{f}(a_m, m_a, b_m, m_b, m_q)$$

- a_m, m_a, b_m, m_b are random bits — changes at every execution even if a and b are fixed.
- So is m_q and q_m
- We can say this an encoded multiplication
- (a_m, m_a) is an encoding of a such that $a = a_m \oplus m_a$
- m_a, m_b, m_q are called masks
- a_m, b_m, q_m are called masked data
    - but this definition will be made more relaxed

The First Example

$$a_m = a \oplus m_a$$

$$b_m = b \oplus m_b$$

$$q_m = q \oplus m_q$$

$$q = f(a, b)$$

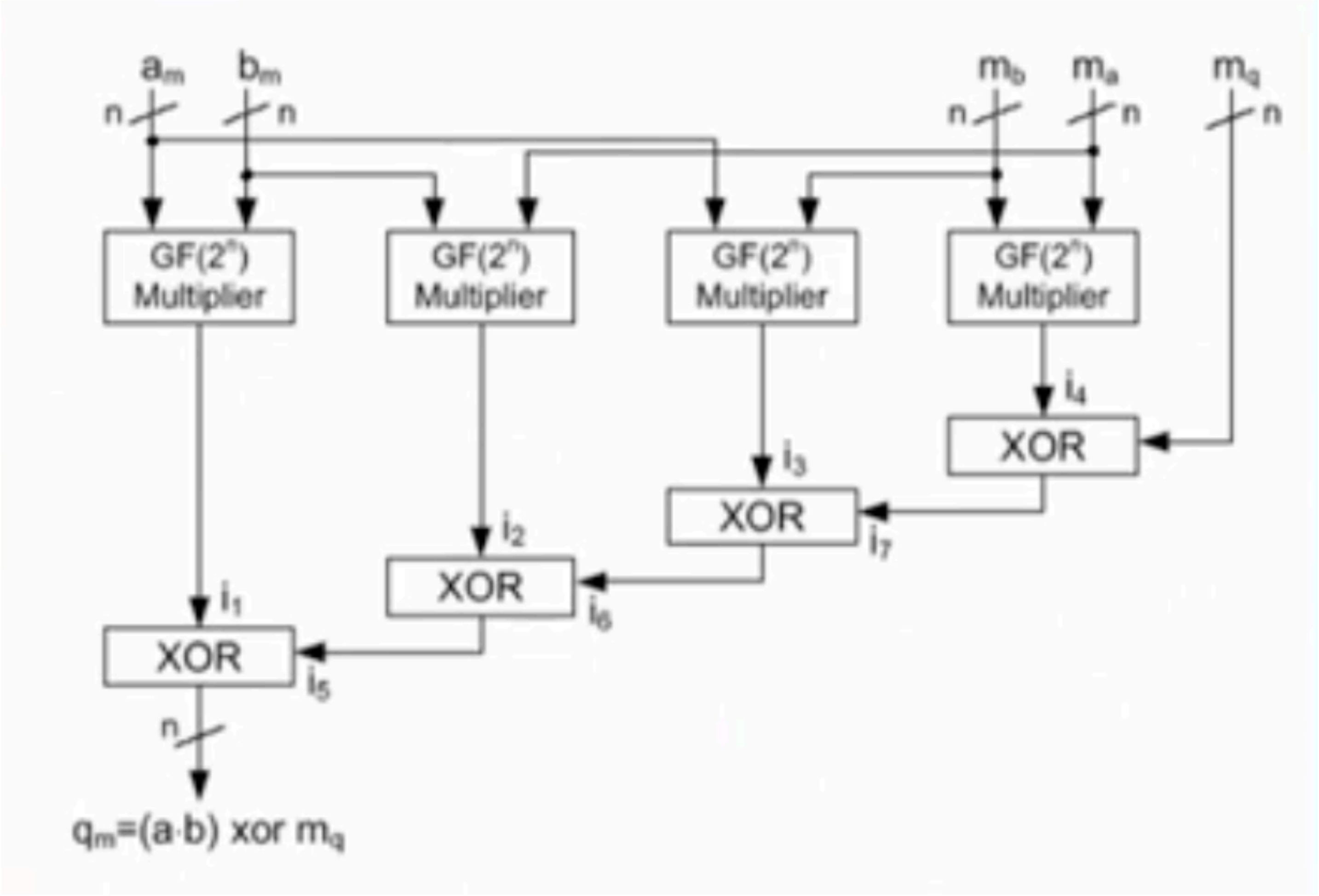
$$q_m = \hat{f}(a_m, m_a, b_m, m_b, m_q)$$

$$\begin{aligned} q_m &= (a \cdot b) \oplus m_q \\ &= (a_m \oplus m_a).(b_m \oplus m_b) \oplus m_q \\ &= (((a_m.b_m \oplus b_m.m_a) \oplus (m_b \cdot a_m)) \oplus m_a.m_b) \oplus m_q \end{aligned}$$

The First Example

- There are 4^5=1024 possible input transmissions that can occur.
- It turns out that the expected value of the energy required for the processing of q=0 and q=1 are identical.
- Thus protected against DPA, under the assumption that the CMOS ***gates switch only once in one clock cycles.***
- But we know there are glitches, and so the output of gates swing a number of times before reaching a steady state. Hence... the argument continues.

Trichina’s Gate



- **Observe**!!! The secret dependent values are always blinded with a mask..
- To realize masked AND, just replace the multipliers with AND gates.

First-Order Analysis

- **The masking here breaks a value X in two *shares* X_1*, and* X_2.**
- $X = X_1 \oplus X_2$
- *Let's consider the leakage as* $l = HW(X_1, X_2)$

x	x_1	x_2	$L(X)$	Mean($L(X)$)	Var($L(X)$)
0	0	0	0	1	1
0	1	1	2		
1	0	1	1	1	0
1	1	0	1		

- First order analysis does not leak.
  - Note the leakage model
- But second order analysis (and higher order) leaks.
- But we can generalise masking…

Higher-Order Attacks

- Like in a 1st order DPA, where we process on a single point on the power trace, in 2nd order attacks, we exploit the joint leakage of two intermediate values that are processed by the device.
- The attack works in the same way as the 1st order attack, except that we preprocess the trace first.
- For example, if we know the points in the trace when f_1 and f_2 are processed, then combining them would reveal information of unmasked data, and then DPA would still work.
- A common preprocessing operation is to take two power values at different times, say p_{t_1}, p_{t_2}, and determine $(p_{t_1} - p_{t_2})^2$

Generalization: compute $(p_{t_1} - \overline{p_{t_1}})(p_{t_2} - \overline{p_{t_2}})$ which works better and is also the second order moment (covariance)

dth-Order Masking

- **The masking here breaks a value X in d+1 *shares* X_1*, and* X_2*.***
- $X = X_1 \perp X_2 \perp \cdots X_{d+1}$
- $\perp$ is some operator — $\oplus$ for operating on gates — Boolean masking
    - Can be an integer addition too, Or some more complex encoding
- Each X_i is a share.
- We operate on the shares
- Ideally, $(d+1)$th order masking should withstand dth order statistical analysis

Computing on Masks: Linear Function

- $X = X_1 \oplus X_2 \oplus \cdots X_{d+1}$

- $lin(X) = lin(X_1 \oplus X_2 \oplus \cdots \oplus X_{d+1}) = lin(X_1) \oplus lin(X_2) \oplus \cdots \oplus lin(X_{d+1})$
- Nonlinear (involving ANDs) is tricky

Computing on Masks: Nonlinear Function

- It is challenging for nonlinear functions.
- Example: $f(X,Y) = Z \oplus XY$
- Masked Circuit:
  - $f_1(X_1, Y_1) = Z_1 \oplus X_1 Y_1$
  - $f_2(X_1, X_2, Y_1, Y_2) = \big((Z_2 \oplus X_1 Y_2) \oplus X_2 Y_1\big) \oplus X_2 Y_2$
- Note again the ordering of the operations is very important!
  - Don't do, $f_2(X_1, X_2, Y_1, Y_2) = (Z_2 \oplus X_1 Y_2) \oplus (X_2 Y_1 \oplus X_2 Y_2)$ …as the second parenthesis is dependent on Y
- However, this is not secure against higher order attacks.
- Actually, not even 1^{st} order attacks if there are glitches.

Masking in Leakage Models

- **Noisy leakage Model:**
  - Each wire leaks its noisy value (independently!!).
  - But then would masking stand?
  - *Noise saves your back…*
  - It has been proven that for distinguishing probability α, masking order *d*, and noise standard deviation σ, the number of observations needed is lower-bounded by $\sigma^{d+4\log\alpha/\log\sigma}$ — that is exponential security amplification!!!
  - But the downside of this model is that you cannot prove security of your masked circuits in real life with this. — two complex to handle the noise etc in formal proofs

Chari S, Jutla CS, Rao JR, Rohatgi P (1999) Towards sound approaches to counteract power-analysis attacks. In: CRYPTO. Lecture notes in computer science, vol 1666. Springer, pp 398–412

Masking in Leakage Models

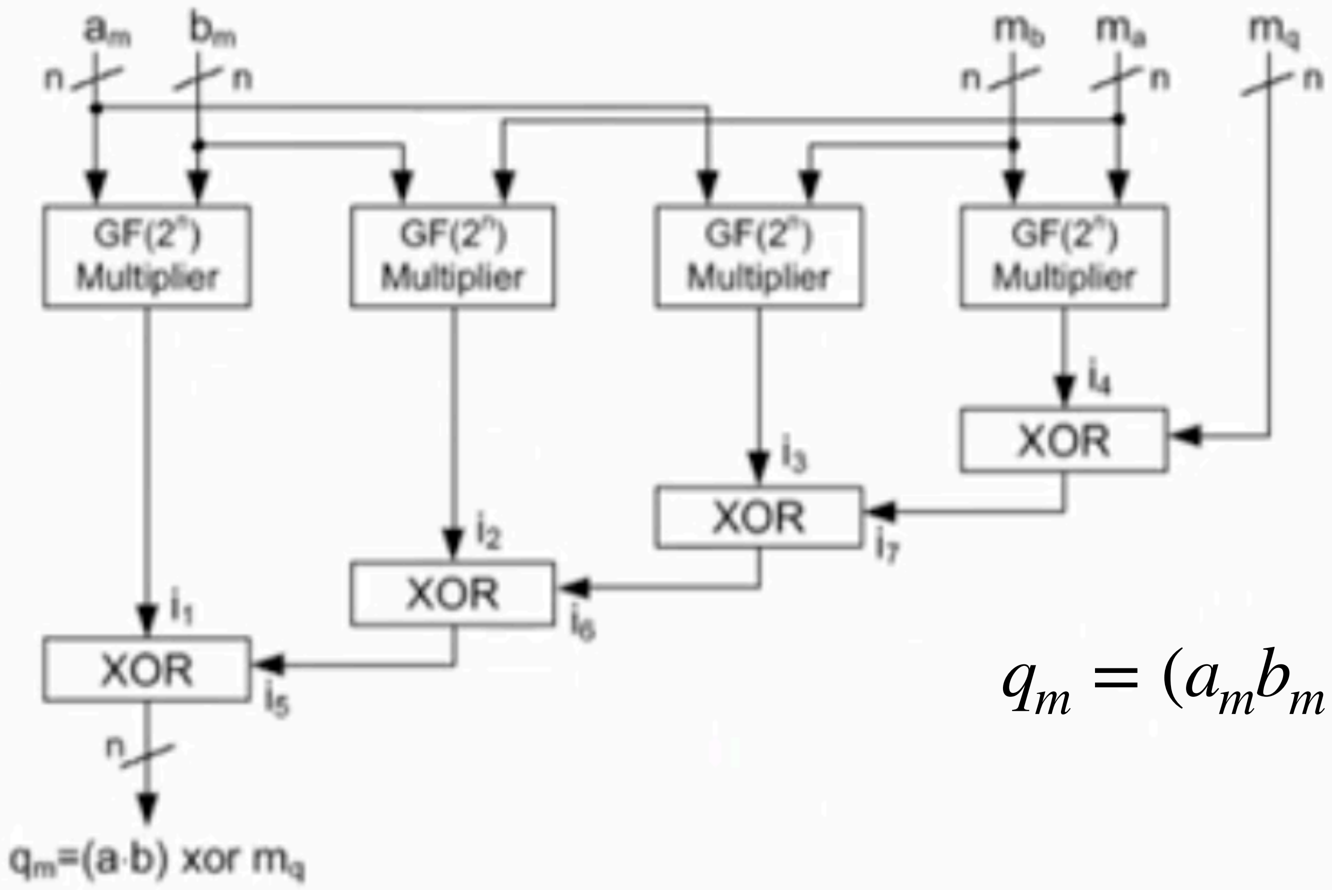
- **Threshold Probing Model:**
  - Upto d wires of adversaries choice leaks their entire value.
  - The choice is not adaptive — that is adversary cannot change the choice per execution.
  - Let us consider a masked circuit with t+1 shares

$$X = X_1 \oplus X_2 \oplus \cdots X_{d+1}$$

  - We prove that for a circuit with n wires, if all possible d-subset of wires are considered one-by-one, **none** of these subsets gives you the actual value of the secret X

Ishai Y, Sahai A, Wagner DA (2003) Private circuits: securing hardware against probing attacks. In: CRYPTO. Lecture notes in computer science, vol 2729. Springer, pp 463–481

Trichina's Gate



- **Observe**!!! The secret dependent values are always blinded with a mask..
- To realize masked AND, just replace the multipliers with AND gates.

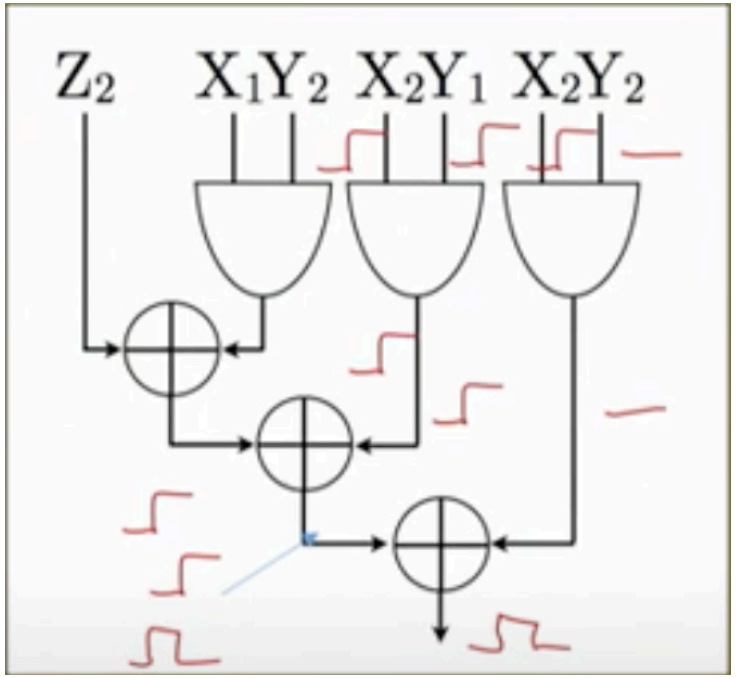
$$q_m = (a_m b_m + (b_m m_a + (m_b a_m + (m_a m_b + m_q))))$$

But Probing Model Does not Capture Reality!!

- **In reality, you leak all wires right?**
- But then why probing model is used??
  - Turns out that you can prove
    - Threshold probing security implies noisy probing security
    - Reductions between the models can be shown
- **Long story short**: You can prove your security in the probing model and it will still remain secure in the noisy probing model with sufficient noise.
- So people go on and keep improving the probing model

Duc A, Dziembowski S, Faust S (2019) Unifying leakage models: from probing attacks to noisy leakage. J Cryptol 32(1):151–177

Security with Glitches: The Glitch Extended Probing Model



y	y_1	y_2	x_2	$z_2 \oplus x_1 y_2$	AND	XOR
0	0	0	0	0	0+0	0+0
0	1	1	0	0	0+0	0+0
0	0	0	1	0	0+0	0+0
0	1	1	1	0	0+2	0+1
0	0	0	0	1	0+0	2+0
0	1	1	0	1	0+0	2+0
0	0	0	1	1	0+0	2+0
0	1	1	1	1	0+2	2+1
1	0	1	0	0	0+0	0+0
1	1	0	0	0	0+0	0+0
1	0	1	1	0	0+1	0+1
1	1	0	1	0	0+1	0+2
1	0	1	0	1	0+0	2+0
1	1	0	0	1	0+0	2+0
1	0	1	1	1	0+1	2+1
1	1	0	1	1	0+1	2+2

Average glitch power for the AND gate does not depend on y.

Average glitch power for the XOR gate depends on y.

- The glitch-extended probing model says that you (adversary) probe a wire, you get the values of all the variables the wire depends on

Plan for Next Class

- **We shall see some really secure masking schemes which are used in practice**

Threshold Implementation

- State-of-the-art masking scheme…
- Glitch-resistant
- Motivated by the idea of Multi-party computation

Threshold Implementation (TI)

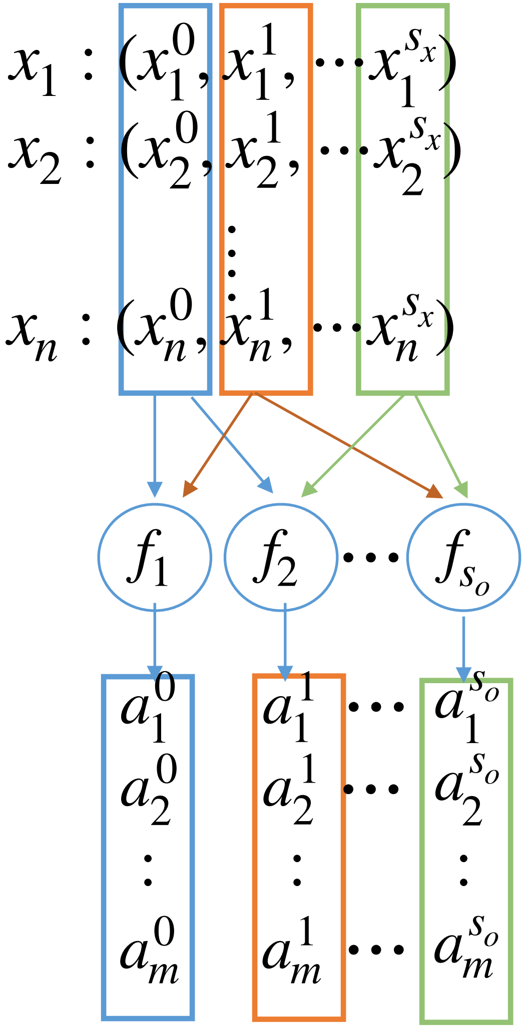
- So far, we have shown how to construct masking and how vulnerable they are to glitches
- But what are the properties that can help in countering glitches?
- TI defines four such properties:
  - Correctness
  - Uniformity
  - Non-completeness
  - Output Uniformity

Threshold Implementation (TI)

- Consider a mapping $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ defined by a vectorial Boolean function $(a_1, a_2, \cdots a_m) = f(x_1, x_2, \cdots x_n)$
- We encode $x_i : (x_i^0, x_i^1, \cdots x_i^{s_x})$ — input shares of the variable x_i
- We now define a vector of functions $\mathbb{f} : (f_1, f_2, \cdots f_{s_o})$
- The output share of variable a_j are denoted as $(a_j^0, a_j^1, \cdots a_j^{s_o})$
- Let' see a pictorial representation…

Threshold Implementation (TI)

$$(a_1, a_2, \cdots a_m) = f(x_1, x_2, \cdots x_n)$$



Correctness

- Let $X_i : (x_i^0, x_i^1, \cdots x_i^{s_x})$ is an encoding of the variable x_i
- Let $A_j : (a_j^0, a_j^1, \cdots a_j^{s_o})$ be an encoding of the variable a_j
- $\mathbb{X} = [X_1, X_2, \cdots X_n]$, and $\mathbb{A} = [A_1, A_2, \cdots A_m]$
- We have $\mathbb{f}(\mathbb{X}) = \mathbb{A}$
- Correctness implies the following:
    - $\forall a \in \mathbb{F}_2^m, \mathbb{f}(\mathbb{X}) = \mathbb{A} \implies \sum_{s_o} a^{s_o} = a = \sum_{s_o} f^{s_o}(\mathbb{x}), \forall \mathbb{x} : \mathbb{x}$ satisfies $\sum_{s_x} x^{s_x} = x \in \mathbb{F}_2^n$
    - Here $\mathbb{x}$ can be seen as a value assignment of $\mathbb{X}$

Uniformity

- Let's for simplicity, denote a random variable X taking value form $\mathbb{F}_2^n$.
- x denote a value assignment to X
- Also, $\mathbb{X}$ is the random variable corresponding to the sharing of X
- We define $Sh(x) = \{\mathbb{x} \in \mathbb{F}_2^{ns_x} | x^1 \oplus x^2 \cdots x^{s_x} = x\}$
- $\mathbb{x}$ is a value assignment to $\mathbb{X}$.
-

Uniform Masking: A masking X is uniform if and only if there exists a constant p such that $\forall x$ we have: if $\mathbb{x} \in Sh(x)$, then $Pr[\mathbb{X} = \mathbb{x} | X = x] = p$, else $Pr[\mathbb{X} = \mathbb{x} | X = x] = 0$, *and* $\Sigma_{\mathbb{x} \in Sh(x)} \Pr[\mathbb{X} = \mathbb{x}] = \Pr[X = x]$

Implication: The value of x is independent of any $s_x - 1$ shares if you have an uniform s_x sharing

Uniformity

- Define, $\mathbb{X}_i$ as the r.v denoting the i^{th} share. Then $\mathbb{X}_{\bar{\imath}}$ denotes the vector without the i^{th} share
- If masking is uniform$\Rightarrow$ $\mathbb{x}_{\bar{\imath}}$ and x are independent for any i.
- $Pr[\mathbb{X} = \mathbb{x} | X = x] = \Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}, \mathbb{X}_i = \mathbb{x}_i | X = x] = \frac{\Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}, \mathbb{X}_i = \mathbb{x}_i, X = x]}{\Pr[X = x]} = \frac{\Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}, \mathbb{X}_i = \mathbb{x}_i, X = x]}{\Pr[X = x, \mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}]} \frac{\Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}, X = x]}{\Pr[X = x]}$

$$= \Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}} | X = x] \Pr[\mathbb{X}_i = \mathbb{x}_i | X = x, \mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}]$$

The last factor equals 1 when $\mathbb{x} \in Sh(x)$ and zero otherwise.

Thus, $\forall x, \Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}} | X = x] = p$.

$\therefore \Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}}] = \Sigma_x \Pr[\mathbb{X}_{\bar{\imath}} = \mathbb{x}_{\bar{\imath}} | X = x] \Pr[X = x] = p = 2^{n(1-s_x)}$

Non-Completeness

- Masked Circuit:
  - $f_1(X_1, Y_1) = Z_1 \oplus X_1Y_1$
  - $f_2(X_1, X_2, Y_1, Y_2) = \big((Z_2 \oplus X_1Y_2) \oplus X_2Y_1\big) \oplus X_2Y_2$
  - Note, f_2 depends on all the 2 shares. Therefore an attacker probing the corresponding wire can observe all the information required.
  - A TI on the other hand ensures that if the attacker probes d wires, it can only provide information for at most $s_{in} - 1$ shares, which is independent of the sensitive data.
- d-th order Non-completeness: Any combination of up to d component functions f_i of $\mathbb{f}$ *must be independent of at least one input share.*

Non-Completeness: Nonlinear Functions

- Let, $(X, Y) \in F_2^2, A = f(X, Y) = XY$.
- Following is a 1st order TI:
  - $A_1 = f_1(X_2, X_3, Y_2, Y_3) = X_2Y_2 \oplus X_2Y_3 \oplus X_3Y_2$
  - $A_2 = f_2(X_1, X_3, Y_1, Y_3) = X_3Y_3 \oplus X_1Y_3 \oplus X_3Y_1$
  - $A_3 = f_3(X_1, X_2, Y_1, Y_2) = X_1Y_1 \oplus X_1Y_2 \oplus X_2Y_1$

Non-Completeness: Affine Functions

- An affine function $f(X) = A$ can be implemented with $s \geq d + 1$ component functions to thwart d-th order attacks.
- Construction:

$$f_1(X_1) = A_1 = f(X_1),$$

For $2 \leq i \leq s$, $f_i(X_i) = A_i$, where f_i is f without constant terms.

  - Eg, $f(X) = 1 \oplus X \Rightarrow f_1(X_1) = 1 \oplus X_1, f_i(X_i) = X_i, 2 \leq i \leq s$

Non-Completeness: The Key Point

If the input mask $\mathbb{X}$ of the shared function $\mathbb{f}$ is a uniform masking and $\mathbb{f}$ is a d-th order TI then the d-th order analysis on the power consumptions of a circuit implementing $\mathbb{f}$ does not reveal the unmasked input value x even if the inputs are delayed or glitches occur in the circuit.

Non-Completeness and Number of Shares

$$S(x,y,z) = x \oplus yz$$
$$= (x_1 \oplus x_2 \oplus x_3) \oplus (y_1 \oplus y_2 \oplus y_3)\,(z_1 \oplus z_2 \oplus z_3)$$

$$S_1(x_2,x_3,y_2,y_3,z_2,z_3) = x_2 \oplus y_2z_2 \oplus y_2z_3 \oplus y_3z_2$$
$$S_2(x_1,x_3,y_1,y_3,z_1,z_3) = x_3 \oplus y_3z_3 \oplus y_3z_1 \oplus y_1z_3$$
$$S_3(x_1,x_2,y_1,y_2,z_1,z_2) = x_1 \oplus y_1z_1 \oplus y_1z_2 \oplus y_2z_1$$

- Here we have first order non-completeness
- **Dependency on the function**: For a function with degree t, you need at least t+1 shares to ensure non-completeness
- So if you want to ensure, **dth order non-completeness**, you need at least **td+1** shares.

Output Uniformity

- Let, $(X, Y) \in F_2^2, A = f(X, Y) = XY$.
- Following is a 1st order TI:
  - $A_1 = f_1(X_2, X_3, Y_2, Y_3) = X_2Y_2 \oplus X_2Y_3 \oplus X_3Y_2$
  - $A_2 = f_2(X_1, X_3, Y_1, Y_3) = X_3Y_3 \oplus X_1Y_3 \oplus X_3Y_1$
  - $A_3 = f_3(X_1, X_2, Y_1, Y_2) = X_1Y_1 \oplus X_1Y_2 \oplus X_2Y_1$

$$X = 0 \Rightarrow X = X_1 \oplus X_2 \oplus X_3 = 0$$
$$Y = 0 \Rightarrow Y = Y_1 \oplus Y_2 \oplus Y_3 = 0$$

Distribution of (A_1, A_2, A_3)

	000	011	101	110
000	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
011	(0,0,0)	(1,1,0)	(1,0,1)	(0,1,1)
101	(0,0,0)	(1,0,1)	(0,1,1)	(1,1,0)
110	(0,0,0)	(0,1,1)	(1,1,0)	(1,0,1)

Output Uniformity

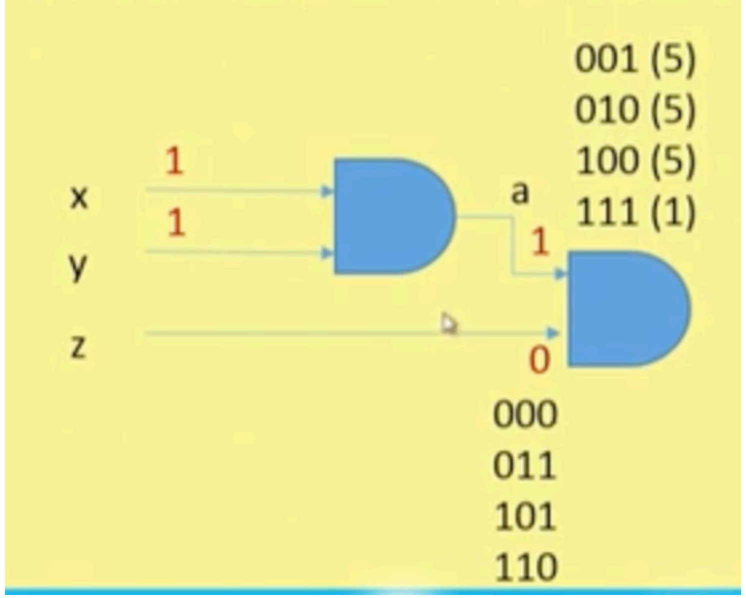
(x, y)	a_1, a_2, a_3 000	011	101	110	001	010	100	111
$(0,0)$	7	3	3	3	0	0	0	0
$(0,1)$	7	3	3	3	0	0	0	0
$(1,0)$	7	3	3	3	0	0	0	0
$(1,1)$	0	0	0	0	5	5	5	1

- This circuit is first order TI, hence secure against first-order attacks
- But what about the case when it is used as input to another circuit.
- Output uniformity becomes crucial…

The average Hamming weights does not depend on (x,y).
For example, if (x,y)=(0,1), average HW=(3x2)x3/16=18/16. If (x,y)=(1,1), average HW=((5x1)x3+3x1)/16=18/16

Output Uniformity

	a_1, a_2, a_3							
(x, y)	000	011	101	110	001	010	100	111
$(0, 0)$	7	3	3	3	0	0	0	0
$(0, 1)$	7	3	3	3	0	0	0	0
$(1, 0)$	7	3	3	3	0	0	0	0
$(1, 1)$	0	0	0	0	5	5	5	1



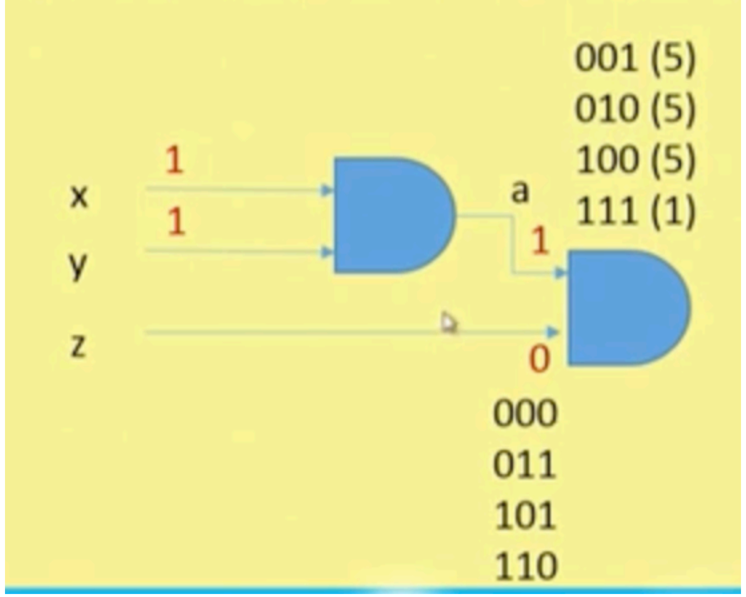
Let’s consider the case with X.Y.Z , where Z is uniformly distributed

000	5+5+5+5+5+5+1=31
011	5+5+1=11
101	5+5+1=11
110	5+5+1=11

Output Uniformity

Let’s consider the case with X.Y.Z , where Z is uniformly distributed

	b_1, b_2, b_3							
(x, y, z)	000	011	101	110	001	010	100	111
$(0,0,0)$	37	9	9	9	0	0	0	0
$(0,0,1)$	37	9	9	9	0	0	0	0
$(0,1,0)$	37	9	9	9	0	0	0	0
$(0,1,1)$	37	9	9	9	0	0	0	0
$(1,0,0)$	37	9	9	9	0	0	0	0
$(1,0,1)$	37	9	9	9	0	0	0	0
$(1,1,0)$	31	11	11	11	0	0	0	0
$(1,1,1)$	0	0	0	0	21	21	21	1



Note, for (x,y,z)=(1,1,0), Average Hamming Weight=11x2x3/64=33/32 ,while for first 6 rows it is 27/32.
These deviations of means with inputs lead to a 1st-order DPA attack.

Uniform Sharing of a Function

- **Uniform Sharing of a Function:** The d-th order sharing $\mathbb{f}$ is uniform if and only if:

$\forall x \in F_2^n, \forall a \in F_2^m, with\ f(x) = a, \forall \mathbb{a} \in Sh(a), and\ s_{out} \geq d + 1:$

$$|\{\mathbb{x} \in Sh(x) \,|\, \mathbb{f}(\mathbb{x}) = \mathbb{a}\}| = \frac{2^{n(s_{in}-1)}}{2^{m(s_{out}-1)}}$$

Uniform Sharing of a Function

- If the masking $\mathbb{X}$ is uniform and the circuit $\mathbb{f}$ is uniform, then masking $\mathbb{A}$ of $a = f(x)$, defined by $\mathbb{A}$= $\mathbb{f}(\mathbb{X})$ is uniform.
- We show: $\Pr([\mathbb{A} = \mathbb{a} | A = a]) = \sum_{\substack{\mathbb{x} \in Sh(x) \\ x, f(x) = a}} \Pr[\mathbb{A} = \mathbb{f}(\mathbb{x}) | A = f(x)] \Pr(\mathbb{X} = \mathbb{x}, X = x)$
- The inner probability in the summation term=$2^{n(s_{in}-1)-m(s_{out}-1)} \Pr(\mathbb{X} = \mathbb{x} | X = x) \Pr[X = x] = 2^{n(s_{in}-1)-m(s_{out}-1)} 2^{-n(s_{in}-1)} = 2^{-m(s_{out}-1)} \Pr[X = x]$
- Thus, we have $\Pr[\mathbb{A} = \mathbb{a} | A = a) = \mathrm{p} = 2^{-m(s_{out}-1)}$, if $\mathbb{a} \in Sh(a)$, and 0 otherwise.

TI of XORs

- XOR is a linear function
- Let $c = a \oplus b$
- Let a_1, a_2, a_3 be the shares of a and b_1, b_2, b_3 be the shares of b i.e.
  - $a = a_1 \oplus a_2 \oplus a_3$
  - $b = b_1 \oplus b_2 \oplus b_3$
- Let c_1, c_2, c_3 be the output shares:
  - $c_1 = a_1 \oplus b_1$
  - $c_2 = a_2 \oplus b_2$
  - $c_3 = a_3 \oplus b_3$
- This is non-complete, uniform and correct
- In fact all the three properties can be achieved using just 2 shares!
- To summarize, TI design for linear functions are EASY!
- XOR + AND is functionally complete, so let us look into TI design of AND gate

TI of ANDs: 2-share is not Enough

- AND is not a linear function
- Let c = ab
- Lets first see whether we can design a 2-share TI
- Let a_1, a_2 be the shares of a and b_1, b_2 be the shares of b i.e.
  - $a = a_1 \oplus a_2$
  - $b = b_1 \oplus b_2$
- The two output shares must contain the following 4 terms:
  - a_1b_1
  - a_1b_2
  - a_2b_1
  - a_2b_2
- In no way can these 4 terms be combined into 2-shares without violating non-completeness.
- So, 2 share TI of AND gate is not possible
- We need to increase the number of shares

TI of ANDs: 2-share is not Enough

- Three shares are good for satisfying non completeness
- But still does not satisfy output uniformity
- We need 4 shares for that!!!

- $A_1 = f_1(X_2, X_3, Y_2, Y_3) = X_2Y_2 \oplus X_2Y_3 \oplus X_3Y_2$
- $A_2 = f_2(X_1, X_3, Y_1, Y_3) = X_3Y_3 \oplus X_1Y_3 \oplus X_3Y_1$
- $A_3 = f_3(X_1, X_2, Y_1, Y_2) = X_1Y_1 \oplus X_1Y_2 \oplus X_2Y_1$

$$A = X.Y$$
$$X = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$
$$Y = y_1 \oplus y_2 \oplus y_3 \oplus y_4$$
$$A = a_1 \oplus a_2 \oplus a_3 \oplus a_4$$
$$a_1 = (x_2 \oplus x_3 \oplus x_4).(y_2 \oplus y_3) \oplus y_3$$
$$a_2 = ((x_1 \oplus x_3).(y_1 \oplus y_4)) \oplus (x_1.y_3) \oplus x_4$$
$$a_3 = (x_2 \oplus x_4).(y_1 \oplus y_4) \oplus x_4 \oplus y_4$$
$$a_4 = (x_1.y_2) \oplus y_3$$

Tricks to Manage Share Explosion

- So if you want to ensure, **dth order non-completeness**, you need at least **td+1** shares.
- Also, to ensure uniformity, you need more shares
- That is bad!!!
- So what to do?

- **Trick 1**: <u>Use fresh randomness</u>

$$A = X.Y$$
$$X = x_1 \oplus x_2 \oplus x_3$$
$$Y = y_1 \oplus y_2 \oplus y_3$$
$$A = a_1 \oplus a_2 \oplus a_3$$
$$a_1 = (x_2.y_2) \oplus (x_2.y_3) \oplus (x_3.y_2) \oplus \mathbf{r_1} \oplus \mathbf{r_2}$$
$$a_2 = (x_3.y_3) \oplus (x_1.y_3) \oplus (x_3.y_1) \oplus \mathbf{r_2}$$
$$a_3 = (x_1.y_1) \oplus (x_1.y_2) \oplus (x_2.y_1) \oplus \mathbf{r_1}$$

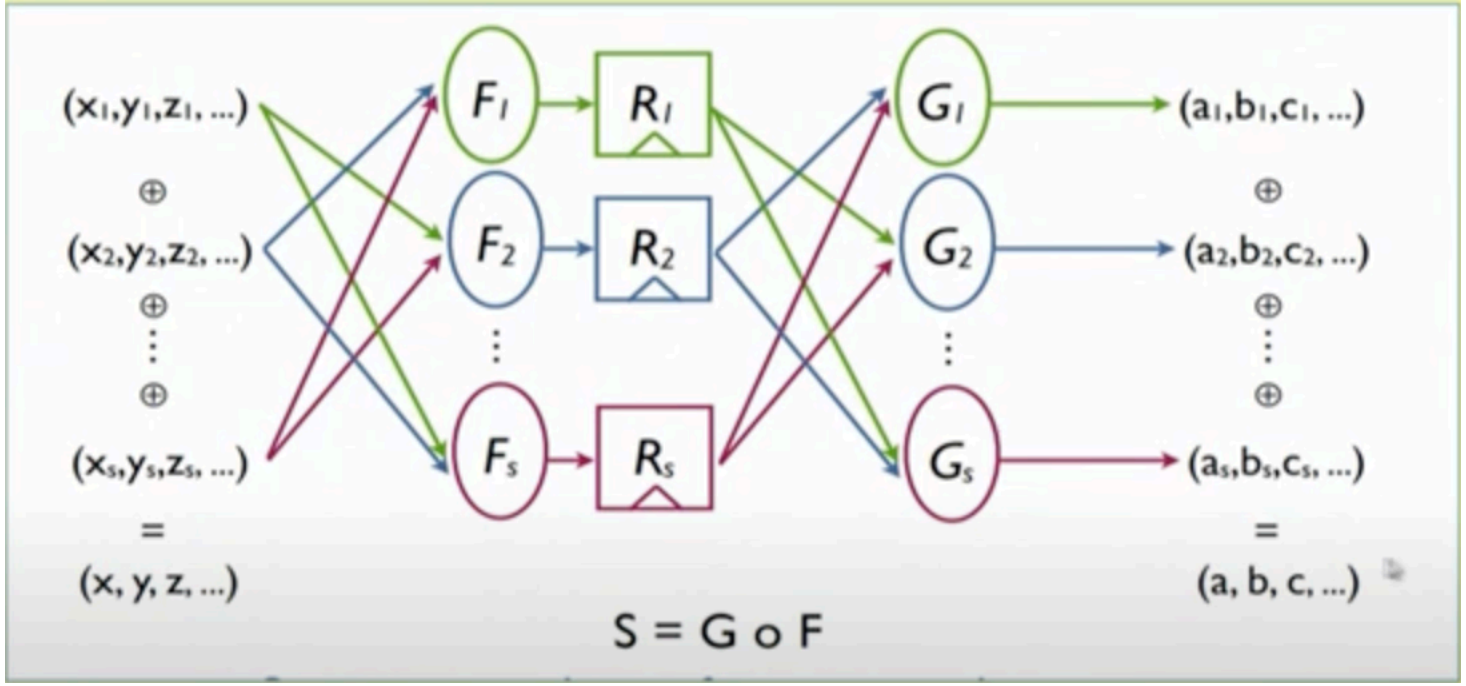
$\mathbf{r_1}$ and $\mathbf{r_2}$ are 2 bits of randomness

$$A = X.Y$$
$$X = x_1 \oplus x_2 \oplus x_3$$
$$Y = y_1 \oplus y_2 \oplus y_3$$
$$A = a_1 \oplus a_2 \oplus a_3$$
$$a_1 = (x_2.y_2) \oplus (x_2.y_3) \oplus (x_3.y_2) \oplus \mathbf{r}$$
$$a_2 = (x_3.y_3) \oplus (x_1.y_3) \oplus (x_3.y_1) \oplus (x_1.\mathbf{r}) \oplus (y_1.\mathbf{r})$$
$$a_3 = (x_1.y_1) \oplus (x_1.y_2) \oplus (x_2.y_1) \oplus (x_1.\mathbf{r}) \oplus (y_1.\mathbf{r}) \oplus \mathbf{r}$$

r is a unit of randomness

Tricks to Manage Share Explosion

- **Trick 2**: **<u>Caution!!! Need Registers…</u>**



- **Why? Because we ensure non-completeness for F and G, through Boolean expressions,**
- **But not for the entire composition — so we need registers.**

Tricks to Manage Share Explosion

- **Trick 2**: Reduce your degree

$$f = XZW \oplus YW \oplus XY \oplus Y \oplus Z$$

$$\begin{aligned}
b_1(X,Y,W) &= X \oplus Y \oplus XW \oplus YW \\
b_2(X,Y,Z) &= Z \oplus XY \oplus XZ \\
b_3(X,Z,W) &= X \oplus W \oplus XZ \oplus ZW \\
f(X,Y,Z,W) &= b_1 \oplus b_2 \oplus b_1 b_3 \oplus b_2 b_3 = b_1(b_1, b_2, b_3)
\end{aligned}$$

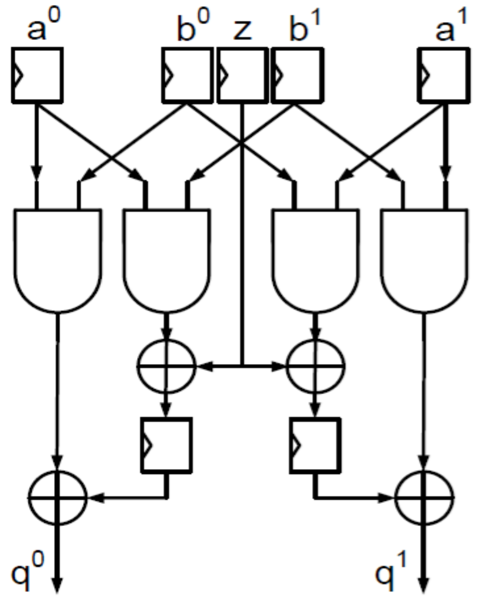
$$\begin{aligned}
b_1 &= b_{11} \oplus b_{12} \oplus b_{13} \\
b_2 &= b_{21} \oplus b_{22} \oplus b_{23} \\
b_3 &= b_{31} \oplus b_{32} \oplus b_{33} \\
X &= X_1 \oplus X_2 \oplus X_3 \\
Y &= Y_1 \oplus Y_2 \oplus Y_3 \\
Z &= Z_1 \oplus Z_2 \oplus Z_3 \\
W &= W_1 \oplus W_2 \oplus W_3
\end{aligned}$$

$$\begin{aligned}
b_{11} &= X_1 \oplus Y_2 \oplus (Y_1W_1) \oplus (Y_1W_2) \oplus (Y_2W_1) \oplus (X_1W_1) \oplus (X_1W_2) \oplus (X_2W_1) \\
b_{12} &= X_2 \oplus Y_3 \oplus (Y_2W_2) \oplus (Y_2W_3) \oplus (Y_3W_2) \oplus (X_2W_2) \oplus (X_2W_3) \oplus (X_3W_2) \\
b_{13} &= X_3 \oplus Y_1 \oplus (Y_3W_3) \oplus (Y_3W_1) \oplus (Y_1W_3) \oplus (X_3W_3) \oplus (X_3W_1) \oplus (X_1W_3) \\
b_{21} &= Z_1 \oplus (Z_1X_2) \oplus (Z_2X_1) \oplus (Y_1X_2) \oplus (Y_2X_1) \oplus (Z_1X_1) \oplus (Y_1X_1) \\
b_{22} &= Z_2 \oplus (Z_2X_3) \oplus (Z_3X_2) \oplus (Y_2X_3) \oplus (Y_3X_2) \oplus (Z_2X_2) \oplus (Y_2X_2) \\
b_{23} &= Z_3 \oplus (Z_1X_3) \oplus (Z_3X_1) \oplus (Y_1X_3) \oplus (Y_3X_1) \oplus (Y_3X_3) \oplus (Z_3X_3) \\
b_{31} &= X_1 \oplus W_2 \oplus (Z_1W_1) \oplus (Z_1W_2) \oplus (Z_2W_1) \oplus (X_1Z_1) \oplus (X_1Z_2) \oplus (X_2Z_1) \\
b_{32} &= X_2 \oplus W_3 \oplus (Z_2W_2) \oplus (Z_2W_3) \oplus (Z_3W_2) \oplus (X_2Z_2) \oplus (X_2Z_3) \oplus (X_3Z_2) \\
b_{33} &= X_3 \oplus W_1 \oplus (Z_3W_3) \oplus (Z_3W_1) \oplus (Z_1W_3) \oplus (X_3Z_3) \oplus (X_3Z_1) \oplus (X_1Z_3)
\end{aligned}$$

Tricks to Manage Share Explosion

- **Trick 3**: **<u>Trick of the day</u>**
  - **Better use of registers**
    - Registers stop glitch propagation, so why not use them as non-completeness tool!!!
    - Note that now we can only use d+1 shares for dth order security, and the degree of the function does not matter anymore

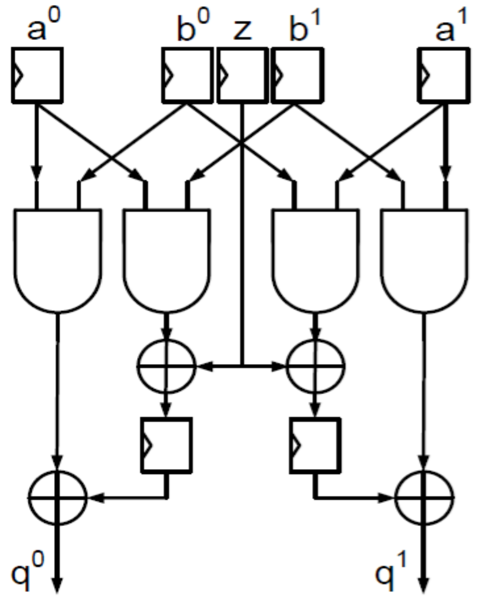
$$q^0 = x^0y^0 + (x^0y^1 + z)$$
$$q^1 = x^1y^1 + (x^1y^0 + z)$$



Link with Probing Model

- **Glitches extend probes**
- **But register and non-completeness stops probe propagation.**
- **So, you can check, if you probe one wire here, you still get no information about the actual secret…**

$$q^0 = x^0y^0 + (x^0y^1 + z)$$
$$q^1 = x^1y^1 + (x^1y^0 + z)$$



Conclusion

- **What we didn't study**
  - More formal treatment of masking in the probing model
  - Instead of output uniformity, probing model has a different way of dealing with composition of masked gates — **composability**
  - **We can prove composability — but requires more detailed formalization**

$$q^0 = x^0y^0 + (x^0y^1 + z)$$
$$q^1 = x^1y^1 + (x^1y^0 + z)$$