DARTH'S SABER
A KEY EXFILTRATION ATTACK FOR SYMMETRIC CIPHERS USING LASER LIGHT

Vittorio Zaccaria, Maria Chiara Molteni, Filippo Melzani, Guido Bertoni

FAULT DIAGNOSIS AND TOLERANCE IN CRYPTOGRAPHY WORKSHOP 2018 - AMSTERDAM NL

Created: 2018-09-12 Wed 22:38

INTRODUCTION

The attack scenario

GOAL OF THIS WORK

- Evaluate the effectiveness of **exfiltrating a key** from a FIA-protected circuit by **injecting double transient faults** using two laser light beams.
- We present some theoretical consideration supported by a quantitative information analysis on an AES implementation.

THE VICTIM CIRCUIT

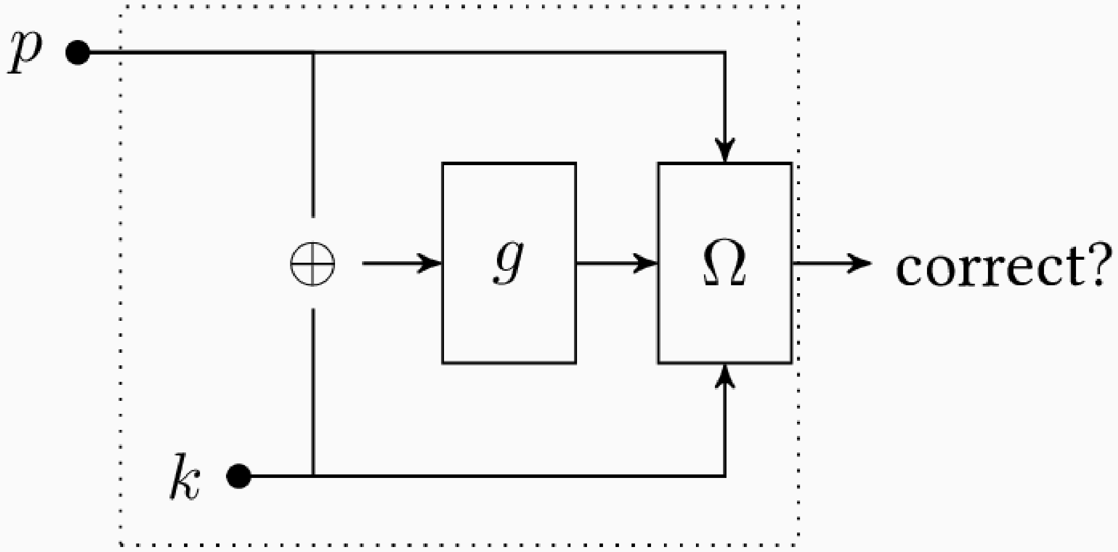

Figure 1: Expected operation of the target device against which the attack will be mounted. k is an unobservable variable within the boundary of the system.

- Assume a circuit computing $g : \mathbb{F}_2^n \to \mathbb{F}_2^m$ that produces **an observable exception** through a FIA mitigation $\Omega : \mathbb{F}_2^m \to \mathbb{F}_2^1$ (e.g., *Boneh et al, Eurocrypt '97*).
- The mitigation produces an exception whenever the result $g(p \oplus k)$ is different from a golden reference $\bar{g}(p \oplus k)$.

DESCRIPTION OF THE ATTACK

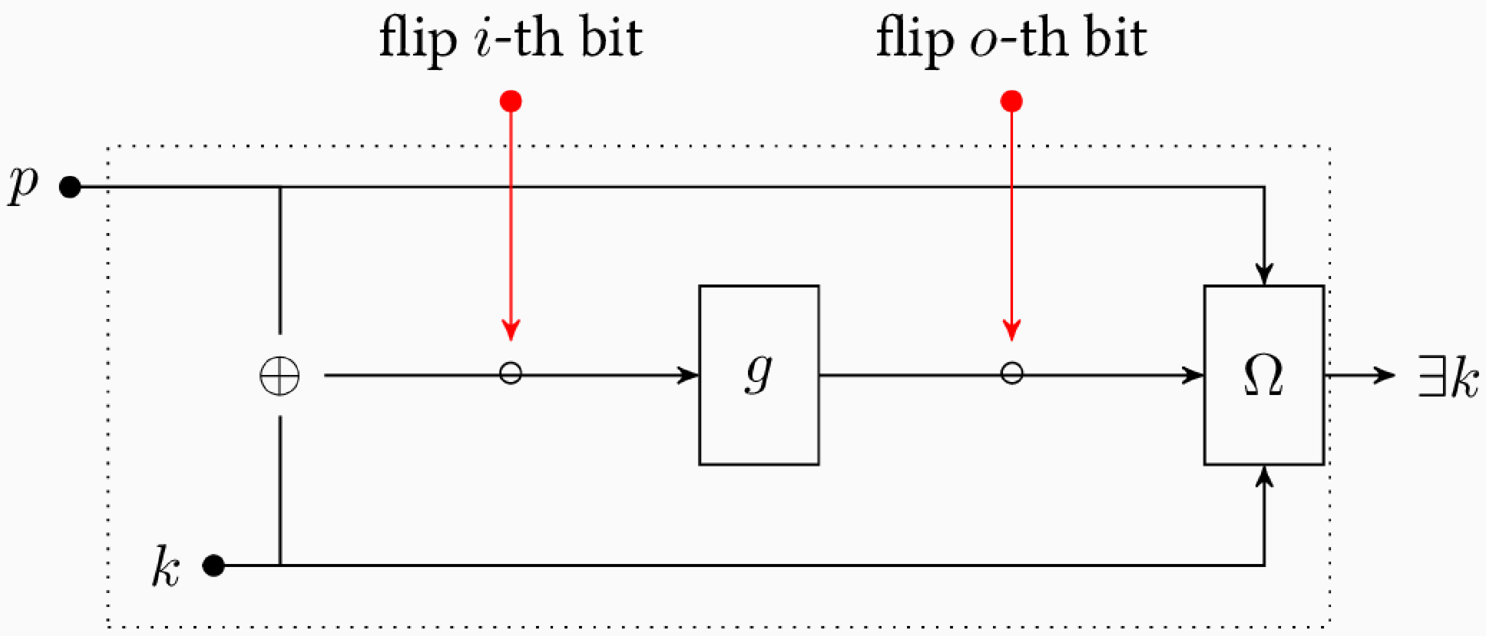

Figure 2: The attack on both input and output buses can be simultaneous or sequential depending on the time it takes to compute g.

- Inject single bit fault over the lines (or registers) that carry $k \oplus p$.
- Later, inject a single bit fault over the lines that carry $g(k \oplus p)$.
- Observe if any exception occurs.

FEASIBILITY OF THE ATTACK

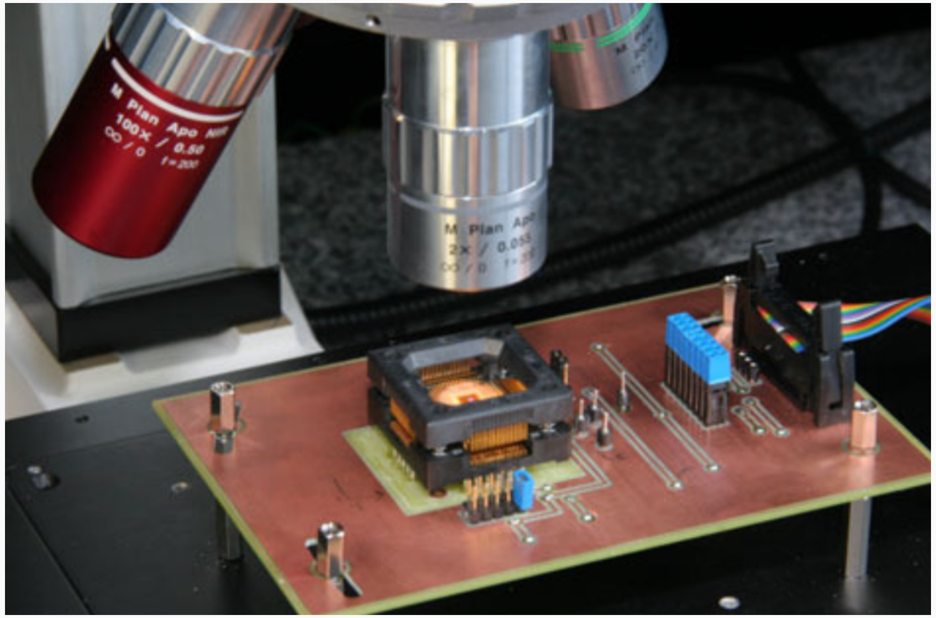

Figure 3: Agoyan et al., *How to flip a bit?*, 2010 IEEE 16th International On-Line Testing Symposium

- In 2010, Agoyan et al. (IOLTS) produced faults in a software AES by targeting SRAM cells; (key take away: SRAM cells easier to attack than chip logic as they are slightly larger).
- Same year, Trichina et al. (FDTC) produced faults in an ARM Cortex M3. SRAM and Flash areas were very difficult to attack (they resolved to attacking a seemingly bus related area).

FORMALIZATION OF THE ATTACK

Supporting concepts

INTERPRETING THE EXCEPTION

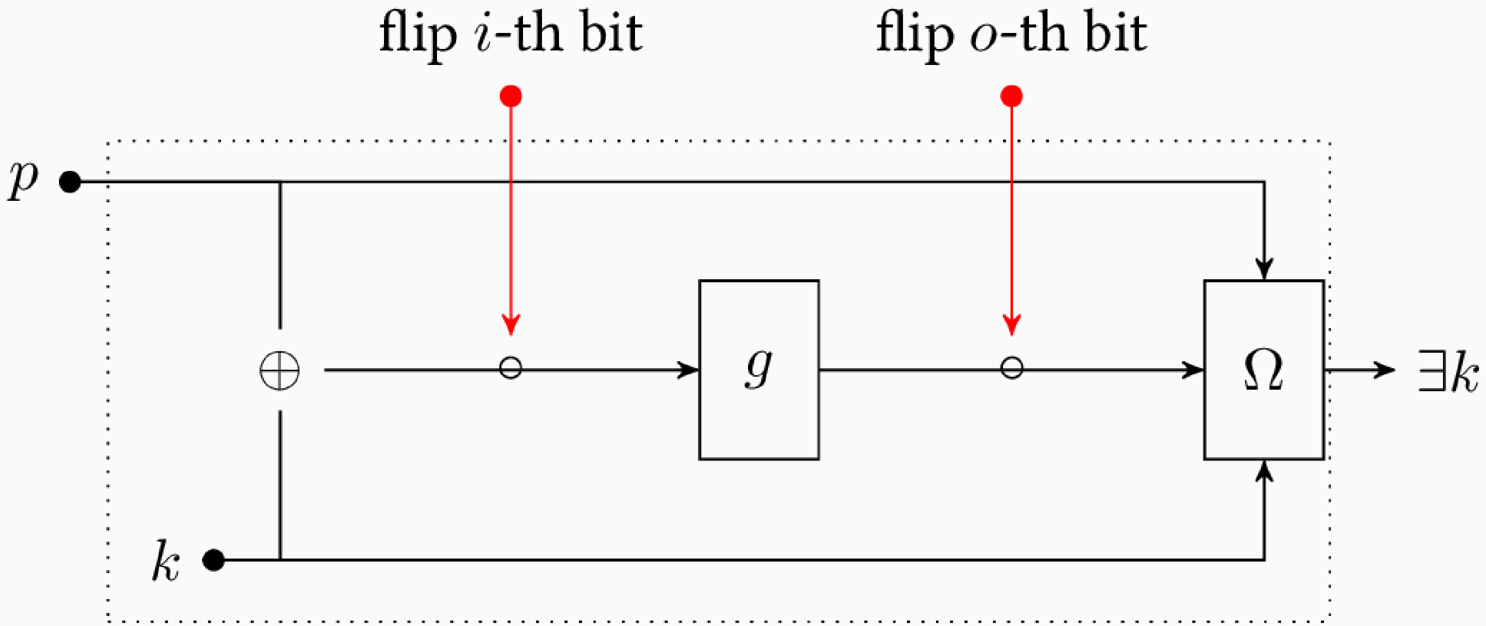

Figure 4: The key concept here is that the mitigation can be seen as the answer from an *oracle* to an existentially quantified boolean predicate.

The exception can be seen as a failed assertion of this boolean predicate:

$$\Omega(p, i, o) = \exists k.\ g((k \oplus p)^{\oplus i})^{\oplus o} == g(k \oplus p)$$

where $x^{\oplus i}$ is value x with the i^{th} bit flipped.

PIVOTAL VARIABLES

Given a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^1$, the i^{th} input variable is **pivotal** iff

$$\exists x.\ f(x) \neq f(x^{\oplus i})$$

where $x^{\oplus i}$ is value x with the i^{th} bit flipped.

An **influencing pair** for $f : \mathbb{F}_2^n \to \mathbb{F}_2^1$ and variable i is a pair $(x, x^{\oplus i})$ witnessing that i is pivotal for f.

$g_0(x) = x_0 \oplus (\neg x_1) \wedge x_2$

x_0	x_1	x_2	g_0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Figure 5: Example. $i = 0$ is **pivotal** with **influencing pair** (001,101)

INFLUENCING SET

An **influencing set** $I_i(f)$ for $f : \mathbb{F}_2^n \to \mathbb{F}_2^1$ and variable i is the quotient set of all influencing pairs for f and i w.r.t. the relation:

$$(x, x^{\oplus i}) \cong (x^{\oplus i}, x)$$

$$g_0(x) = x_0 \oplus (\neg x_1) \wedge x_2$$

x_0	x_1	x_2	g_0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$(0, -, 1)$

$(1, -, 1)$

Figure 6: Example: $i = 1$ has an influencing set composed of only two pairs $(0-1, 1-1)$.

MULTIPLE OUTPUT FUNCTIONS

Given a function $g : \mathbb{F}_2^n \to \mathbb{F}_2^n$ we might want to view it as a set of n functions $\{g_o : o \in [1, n]\}$ of the type $\mathbb{F}_2^n \to \mathbb{F}_2^1$ so as to characterize each output bit of g with its own set of influencing pairs.

Clearly, we are interested in unique influencing pairs that **change only one output**. Enter the **reduced influencing set**.

REDUCED INFLUENCING SET (RIS)

A **reduced influencing** set $R_i(g_o)$ is the set $I_i(g_o)$ where pairs present in other output bits have been removed, i.e.,

$$R_i(g_o) = I_i(g_o) - \bigcup_{j \neq o} I_i(g_j)$$

Note that any set $R_i(g_o)$ is exactly the set of values for which the oracle Ω gives a positive answer.

$(0,0,-)$ $(1,0,-)$ $I_2(g_0)$

$$g_0(x) = x_0 \oplus (\neg x_1) \wedge x_2$$
$$g_1(x) = \neg x_0 \wedge \neg x_1 \wedge \neg x_2$$

$(0,0,-)$ $I_2(g_1)$

$$R_2(g_0) = I_2(g_0) - I_2(g_1) = \{(1,0,-)\}$$
$$R_2(g_1) = I_2(g_1) - I_2(g_0) = \{\}$$

Figure 7: Example **reduced influencing set** for a function $g(x) = [g_0(x), g_1(x)]$.

EXAMPLE: RIS

For the vector function $g : \mathbb{F}_2^3 \to \mathbb{F}_2^2, g(x) = [g_0(x), g_1(x)]$ we get the reduced influencing sets shown on the right.

For example, if we manage to get a positive answer when attacking (x_1, g_1) we might recover the entire set of input values.

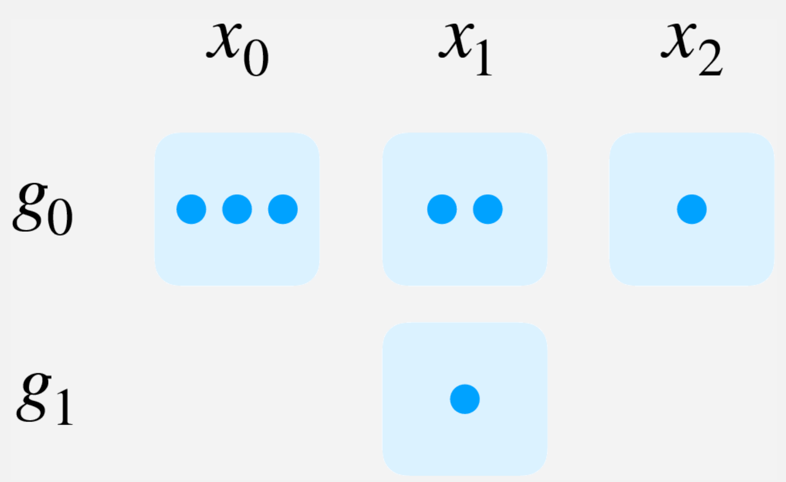

Figure 8: Example: size of influencing sets for the considered multi-output function $g = [g_0, g_1]$

DERIVABLE INFORMATION

Let us consider i and o fixed; in principle, a positive answer from the oracle provides an amount of self-information on the input equivalent to

$$\alpha(i, o) = -\log_2 \frac{\rho}{2^{n-1}} \quad \text{where } \rho = |R_i(g_o)|$$

Instead, the information quantity associated with a negative answer is:

$$\omega = -\log_2 \frac{2^{n-1} - \rho}{2^{n-1}}$$

HOW TO EXPLOIT INFORMATION

We can produce an average information measure for both negative and positive answers given by using the binary entropy function H_g:

$$H_g(i, o) = p\alpha + (1 - p)\omega \ \text{ where } \ p = \frac{\rho}{2^{n-1}}$$

The binary entropy function can guide the attacker in identifying the most ``informing'' input and output/combinations; in principle, one would want to **investigate combinations $(\boldsymbol{i}, \boldsymbol{o})$ that have highest entropy**, as the less entropic ones might provide higher self-information less frequently.

EXAMPLE: COMPUTING INFORMATION

Considering the previous example, we get a binary entropy:

	x_0	x_1	x_2
g_0	0.81	1	0.81
g_1	0	0.81	0

In turn, this suggests to bit-flip the second input variable and the output of the first function g_0 in order to obtain the maximum information.

EXAMPLE APPLICATION TO UNMASKED AES

Is there exploitable entropy within the SBox for this attack?

SELF-INFORMATION (SBOX)

A simulation analysis shows that there are only 24 combinations (among 64) that provide an entropy $h = 0.0659$ different from 0:

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
g_0	•		•	•				•
g_1			•			•	•	•
g_2						•		
g_3	•				•			
g_4	•			•			•	
g_5		•	•			•		•
g_6	•				•	•		
g_7		•		•				•

• $\rightarrow h = 0.0659$

The entropy is very small, but the self-information associated with a positive outcome is 7 as each non-null entropy point corresponds to **a reduced influencing set composed of a single influencing pair**.

DESCRIPTION OF THE ATTACK

A practical attack to a single SBOX would go as follows;

1. An attacker selects a plain-text $\bar{p}$ and an input/output pair (i, o) among the 24 with non-null entropy.
2. She then triggers encryption by injecting faults and observes if the system generates any exception.
3. Assume no exception is raised; then the following predicate is true:

$$\exists k.\, \mathrm{SBOX}((k \oplus \bar{p})^{\oplus i})^{\oplus o} == \mathrm{SBOX}(k \oplus \bar{p})$$

DERIVING THE KEY

To derive the key, we recall that the reduced influencing set for any non-null entropy pair (i, o) of the SBOX is **composed by a single influencing pair $\boldsymbol{x}$**.

This means that either $k \oplus \bar{p} = x$ or $k \oplus \bar{p} = x^{\oplus i}$, i.e, for each bit j of the key we have

$$k_j = x_j \oplus \bar{p}_j, j \in [0 \dots 7] \wedge j \neq i$$

COMPUTATIONAL COMPLEXITY

- Attacker must evaluate at most 2^7 values for $\bar{p}$ as she knows that the i^{th} bit does not influence the exception generation.
- By consequence, to derive the 112 bits from a 128 bit key, the attacker has to perform 2048 injections (worst case).

EXAMPLE APPLICATION TO MASKED AES

Is an SCA-protected AES vulnerable to this attack?

MASKED IMPLEMENTATION (1ST ORDER PROTECTION)

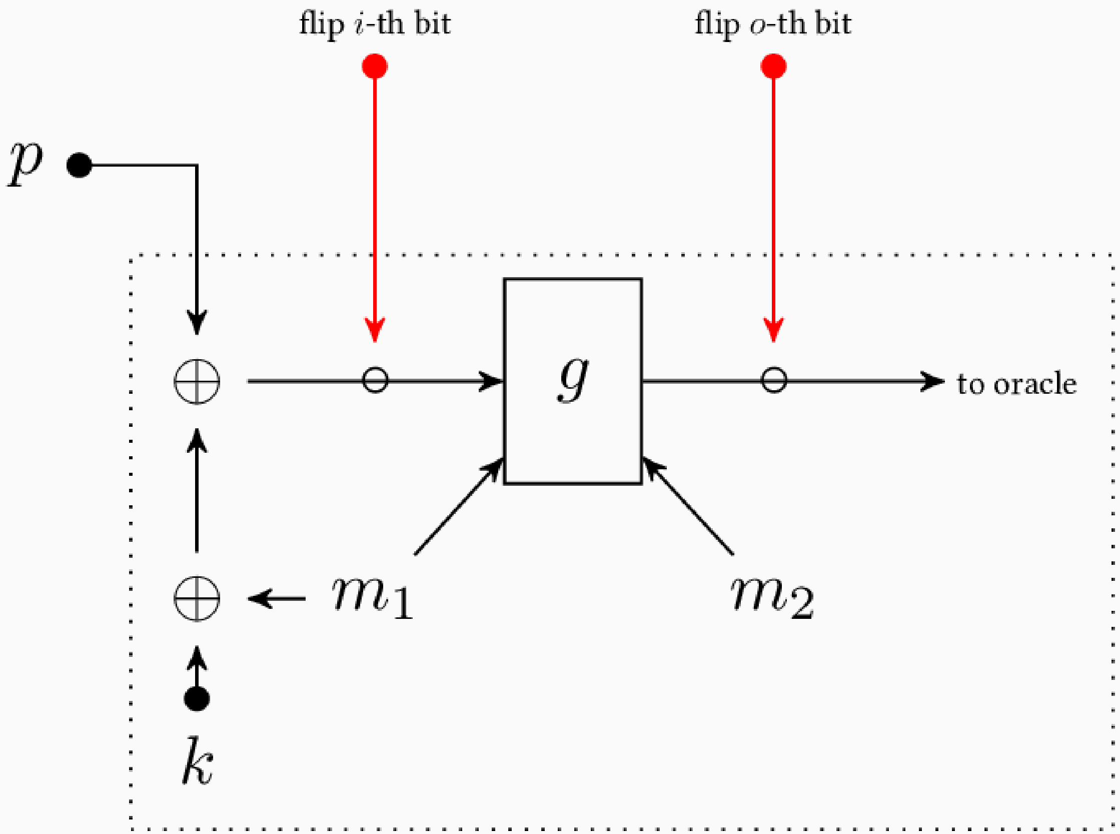

Figure 11: In a masked implementation, a random mask m_j is always added to the secret value, either an input or an output of g.

TRANSPARENCY?

No exception would correspond to the following satisfiability condition:

$$\begin{aligned}\Omega^*(p, i, o) =& \exists k m_1 m_2 . \\ & g((k \oplus p \oplus m_1)^{\oplus i}, m_1, m_2)^{\oplus o} == \\ & g(k \oplus p \oplus m_1, m_1, m_2) \\ =& \exists k \\ & \mathrm{SBOX}((k \oplus p)^{\oplus i})^{\oplus o} == \\ & \mathrm{SBOX}(k \oplus p)\end{aligned}$$

The latter being equivalent to the original oracle, we would obtain the same amount of information regardless of masking.

COUNTERMEASURES

How to prevent this attack

COUNTERMEASURES / MAKE IT SMARTER

- Attacker has a relatively low probability of having a favourable result as she needs to inject many faults for retrieving the key.
- The device could detect this anomalous situation and reduce the amount of information provided back to the user; e.g., **silencing exceptions randomly**.

CONCLUSIONS

Have we learned something?

WHAT HAVE WE LEARNED

- It is possible to exploit a double fault at the input and output of a function and exploit fault attack mitigations to still get information out of it
- It is possible to make some practical consideration in the case of AES.
- Masking seems not an issue for attackers willing to use this method.

THANK YOU