A note on the security of Higher-Order Threshold Implementations*

Oscar Reparaz
oscar.reparaz@esat.kuleuven.be

KU Leuven/COSIC and iMinds

Abstract. At ASIACRYPT 2014, Bilgin et al. describe higher-order threshold implementations: a masking countermeasure claiming resistance against higher-order differential power analysis attacks. In this note, we point out that higher-order threshold implementations do not necessarily provide higher-order security. We give as counterexamples two concrete higher-order threshold implementations that exhibit a second order flaw.

1 Higher-Order Threshold Implementations

We refer the reader to [1] for background information. Higher-order threshold implementations (HOTI) have the remarkable property of not needing extra randomness during computation, if each sharing (=masked function) satisfies some properties (namely, uniformity). (This extra randomness is usually called refreshing in other publications.) In the rest of this note, we show that higher-order security is not preserved through the composition of arbitrary yet uniform sharings. Thus, it is possible to conceive HOTI designs that are not higher-order secure.

2 A counterexample

For the sake of simplicity, in this example we compute on 1 unshared bit, split in 5 shares (one bit per share). We aim at second order security. We define the sharing $F_{i,j,k}$ for the identity function as follows: on input $(a_1, \ldots, a_5) \in \mathbf{F}_2^5$ it outputs $(b_1, \ldots, b_5) \in \mathbf{F}_2^5$ as $b_m = a_m$ for $m \notin \{j, k\}$, $b_j = a_j + a_i$ and $b_k = a_k + a_i$. This sharing is parametrized by the tuple (i, j, k). A concrete instantiation of the sharing is for example $F_{1,2,3}(a_1, a_2, a_3, a_4, a_5) = (a_1, a_2 + a_1, a_3 + a_1, a_4, a_5)$ as shown in Fig. 1. The reader can verify that $F_{i,j,k}$ is correct (it computes the identity function), second order non-complete and uniform (as long as i, j, k are all different.)

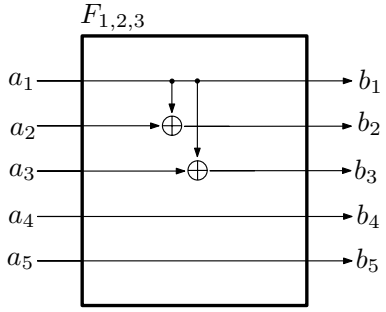

Fig. 1. The sharing $F_{1,2,3}$.

Consider now the composition $F_{5,1,2} \circ F_{4,1,2} \circ F_{2,1,3}$, that is, first applying $F_{2,1,3}$ on the input shares $(a_1, a_2, a_3, a_4, a_5)$, then applying $F_{4,1,2}$ on the output of the first function and finally applying $F_{5,1,2}$. This composition is shown in Fig. 2. Let us write the intermediate variables:

* Version as of 2014.12.10. The latest version is at http://www.esat.kuleuven.be/~oreparaz/hoti/.

$$
\begin{array}{llll}
(a_1, & a_2, & a_3, & a_4, a_5) \quad \text{input} \\
(a_1 + a_2, & a_2, & a_3 + a_2, & a_4, a_5) \quad \text{output after } F_{2,1,3} \\
(a_1 + a_2 + a_4, & a_2 + a_4, & a_3 + a_2, & a_4, a_5) \quad \text{output after } F_{4,1,2} \\
(a_1 + a_2 + a_4 + a_5, & a_2 + a_4 + a_5, & a_3 + a_2, & a_4, a_5) \quad \text{output after } F_{5,1,2}
\end{array}
$$

Then the pair of variables (v_1, v_2) with $v_1 = a_3$ (third share of input) and $v_2 = a_1 + a_2 + a_4 + a_5$ (first share of the output after $F_{5,1,2}$) is not independent from the unshared variable $a_1 + a_2 + a_3 + a_4 + a_5$. Thus, this particular valid HOTI construction is second order insecure.

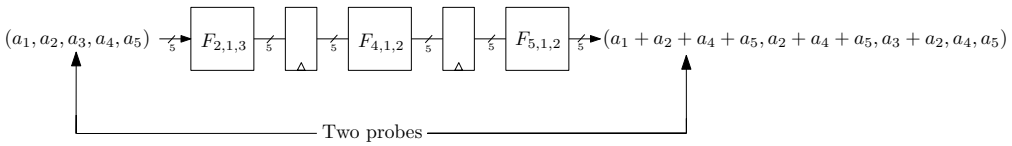

Fig. 2. The composition $F_{5,1,2} \circ F_{4,1,2} \circ F_{2,1,3}$ with input $(a_1, a_2, a_3, a_4, a_5)$, showing the two values (v_1, v_2) that exhibit second order leakage

This particular HOTI construction is only an example. Instead of using $F_{i,j,k}$, one could choose a more straightforward sharing such that one attains higher-order security. However, the point here is that while correctness, higher-order non-completeness and uniformity may provide security *locally*, by themselves they do not provide higher-order security *globally*, that is, when the adversary is allowed to place probes in different functions.

3 A non-linear example

Consider the following NLFSR. The unmasked state consists of 4 bits: $L[0]$ to $L[3]$. Taps are at indices $1, 2, 3$, feedback is plug at position 0. The feedback function $f = f(L[3], L[2], L[1])$ is the AND-XOR function as of Eq. 5 from [1]. The shared version follows the lines of [1]; in particular, $L[0]$ is split into 10 shares and $L[1], L[2], L[3]$ are split into 5 shares. The conversion from 10 to 5 shares is done as in [1]. In particular, the 5th share of $L[1]$ sees 5 shares of $L[0]$ when the cipher is clocked.
The reader can verify that the 5th share of $L[1]$ at the end of the second cycle and the 5th share of $L[1]$ at the end of the tenth cycle leak jointly information about the initial state in the second statistical order.

4 Mitigation

One possible mitigation is to refresh the shares after each computation, for example by adding fresh shares of the null vector. The idea here is to isolate the intermediates occuring within each computation stage from intermediates of another stage, so that combining intermediates from different stages no longer reveals information about a secret intermediate. The exact amount of refresh blocks needed to make the implementation higher-order secure may depend from design to design. This fix naturally increases area and randomness requirements.

Acknowledgements. The author thanks the authors of [1] and Fré Vercauteren for interesting discussions. The author is supported by a PhD fellowship from the Fund for Scientific Research - Flanders (FWO).

References

1. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen. Higher-Order Threshold Implementations. ASIACRYPT 2014.