SNI-in-the-head: Protecting MPC-in-the-head Protocols against Side-channel Analysis

Okan Seker, Sebastian Berndt, and Thomas Eisenbarth

University of Lübeck, Germany
{okan.seker,s.berndt,thomas.eisenbarth}@uni-luebeck.de

Abstract. MPC-in-the-head based protocols have recently gained much popularity and are at the brink of seeing widespread usage. With this widespread use come the spectres of implementation issues and implementation attacks. Side-channel attacks are a serious threat to the security of implementations of secure cryptographic protocols due to unintended leakage of sensitive information. We show that implementations of protocols constructed by the MPC-in-the-head paradigm are vulnerable to such attacks. As a case study, we choose the ZKBoo-protocol of Giacomelli, Madsen, and Orlandi (USENIX 2016) and show that even a single leaked value is sufficient to break the security of the protocol. To show that this attack is not just a theoretical vulnerability, we apply differential power analysis to show the vulnerabilities of the device.
In order to remedy this situation, we extend and generalize the ZKBoo-protocol by making use of the notion of strong non-interference of Barthe et al. (CCS 2016). To apply this notion to ZKBoo, we construct novel versions of strongly non-interfering gadgets that balance the randomness across the different branches evenly. Finally, we show that each circuit can be decomposed into branches using only these balanced strongly non-interfering gadgets. This allows us to construct a version of ZKBoo, called $(n+1)$-ZKBoo secure against side-channel attacks with very limited overhead in both signature-size and running time. Furthermore, $(n+1)$-ZKBoo is scalable to the desired security against adversarial probes. We experimentally confirm that the attacks successful against ZKBoo no longer work on $(n+1)$-ZKBoo.

Keywords: MPC-in-the-head, Zero Knowledge, Strong Non-Interference

1 Introduction

Multiparty computation (MPC) is one of the most widely studied cryptographic primitives. For a long time, MPC-protocols were believed to be of purely theoretic interest, but recent developments have shown that they are usable for a wide range of practically relevant applications. One well-studied version of these protocols, called *MPC-in-the-head*, is on the brink of seeing widespread use, e. g. as part of NIST's Post-Quantum Signature Suite of algorithms in the form of Picnic [CDG$^+$17]. Due to the development of practical MPC protocols, efficient implementations of such protocols were also developed. In this paper, we study implementation attacks (in the form of side-channel attacks) against MPC protocols.

Side-channel attacks aim to use weaknesses in the implementation of a cryptographic protocol rather than trying to break the protocol itself. The main idea behind these attacks is to use physical information given by the computer on which the implementation of the protocol is running such as timing information [Koc96], cache behavior [TOS10,YF14], or power consumption [KJJR11]. These attacks have been successfully performed for over 25 years and many mitigation plans have been devised for them. As these plans often come at the cost of the running time or memory of the protocol, many implementations are still vulnerable to them. In this work, we focus on *differential power analysis* (DPA) attacks [KJJ99] that measure the power consumption during the run of an implementation and use statistical tools to derive information about the value of variables at certain points in time. While DPA attacks are known to be widely applicable against many implementations, mitigation plans have also been studied intensively.

Many public-key protocols used in modern systems such as RSA or the Diffie-Hellman key exchange have shown to withstand attacks run on classical computers. On the other hand, it is long known that attackers equipped with quantum computers are provably able to break these systems [Sho99]. While the deployment of practical quantum computers seems to be unlikely within the next decade, the cryptographic community has already started to develop algorithms that withstand these quantum attackers. To standardize some of these algorithms and to foster their timely adoption, NIST has started their *Post-Quantum Cryptography Standardization* project in 2017, where 82 algorithms were submitted, 69 proceeded to the first round and 26 to the second round [CCJ$^+$16,NIS20a,NIS20b]. The standardization focuses on key encapsulation mechanisms (KEMs), from which public-key encryption schemes can be derived easily, and on signature schemes due to their widespread use in modern systems. While the security of most of these algorithms relies on problems revolving around lattices or coding theory that are believed to be intractable for quantum computers, the signature scheme Picnic [CDG$^+$17] relies on the hardness of SHA-3 and a low-complexity cipher called LowMC [ARS$^+$15]. The underlying zero-knowledge protocol of Picnic follows the MPC-in-the-head paradigm and it is called ZKB++ which is an optimised version of ZKBoo [GMO16]. We will use the ZKBoo protocol as an example both for our attacks on the general MPC-in-the-head paradigm as well as for our defence mechanisms.

Our results

In this work, we study the applicability of DPA attacks to procotols relying on the MPC-in-the-head paradigm. We also show how one can prevent such attacks. To show the versatility of our approach, we use the ZKBoo protocol [GMO16] as an example.

As a first step, we show that ZKBoo is vulnerable to DPA attacks that extract only a single variable, as the protocol itself reveals two out of three shares. To show that this line of attack is feasible, we impement a DPA adversary that is able to recover the third share with a high degree of certainty. In order to remedy such attacks, we first generalize the notion of $(2, 3)$-decompositions of functions introduced in [GMO16]. This allows us to apply MPC-based *masking techniques*—which are widely used to counteract DPA attacks—in the setting of MPC-in-the-head protocols. To use MPC-based DPA protection into MPC-in-the-head protocols, we need gadgets that on t probes only reveal at most t inputs. Formally, this requirement is captured by the notion of *strong non-interference* (SNI) [BBD$^+$16]. While SNI gadgets are known in the literature, none of them are compatible with the function decompositions needed for ZKBoo, as the dependency between the partial functions is imbalanced. Hence, we design suitable *balanced* SNI gadgets to obtain a generalized version of ZKBoo, called $(n+1)$-ZKBoo that reveals $\lceil n/2\rceil + 1$ shares out of $n+1$. Any attacker obtaining $n - (\lceil n/2\rceil + 1)$ additional variables is still not able to recover the complete input. To show the feasibility of our defense mechanisms, we implemented this algorithm for $n + 1 = 5$ (thus revealing three shares). Our experiments show that the extraction of a single additional variable is not sufficient to reconstruct the input. To protect against n probes, the size of the communication of $(n+1)$-ZKBoo is about $(n + 1)/4$ times larger than those of the original ZKBoo, while the running time of $(n + 1)$-ZKBoo is about $(n + 1)(n + 2)/9$ times larger than ZKBoo.

2 Preliminaries

First, we summarize the notion used in the rest of the paper. In the following, we fix some finite ring $(\mathbb{K}, \oplus, \otimes)$ with an *addition* operation $\oplus$ and a *multiplication* operation $\otimes$. As usual, we often omit the multiplication symbol $\otimes$ and thus write xy instead of $x \otimes y$. For $a, b \in \mathbb{Z}$ with $a < b$, we define $[a, b] := \{a, a + 1, \dots, b - 1, b\}$. The letters $x, y, z, \dots$ represent the sensitive variables. Random variables are represented by a letter r, with an index as r_i. To denote a random selection of a variable r from the field $\mathbb{K}$, we use $r \in_R \mathbb{K}$.

Typically, a variable x is split into $n + 1$ shares $x_0, \dots, x_n$ such that $x = \bigoplus_{i=0}^{n} x_i$. The value n is called the *masking order*. This technique of masking was popularized in [CJRR99a]. A vector of shares $(x_0, \dots, x_n)$ is denoted by $\overline{x}$, and the underlying masked value is given by $x = \bigoplus_{i=0}^{n} x_i$. For a subset $I \subseteq [0, n]$ of indices, we denote by $x_{|I} = (x_i)_{i\in I}$ the sub-vector of shares indexed by I. A *gadget* G for a function $f\colon \mathbb{K}^a \to \mathbb{K}^b$ (with regard to a masking order) is an arithmetic circuit as a combination of d gates with $a \cdot (n + 1)$ inputs and $b\cdot(n+1)$ outputs grouped into a vectors of shares $\overline{x}^{(1)}, \dots, \overline{x}^{(a)}$, resp. b vectors of shares $\overline{y}^{(1)}, \dots, \overline{y}^{(b)}$. The arithmetic circuits have five kinds of gates: the unary $\oplus_\alpha$ gate with $\alpha \in \mathbb{K}$, which on input x outputs $x \oplus \alpha$; the unary $\otimes_\alpha$ gate with $\alpha \in \mathbb{K}$, which on input x outputs $x \otimes \alpha$; the binary $\oplus$ gate which on inputs x, x' outputs $x \oplus x'$; the binary $\otimes$ gate, which on inputs x, x' outputs $x \otimes x'$;

and the random gates with fan-in 0 that produce a uniformly chosen random element $r \in_R \mathbb{K}$. Note that in the case of $\mathbb{K} = \mathrm{GF}(2)$, these gates directly correspond to AND, XOR, and NOT gates. The gadget needs to be correct, i. e. $G(\overline{x}^{(1)}, \ldots, \overline{x}^{(a)}) = (\overline{y}^{(1)}, \ldots, \overline{y}^{(b)})$ iff $f(x^{(1)}, \ldots, x^{(a)}) = (y^{(1)}, \ldots, y^{(b)})$ for all possible inputs and for all values generated by the random gates. The values assigned to wires that are not output wires are called *intermediate variables*.

We also make use of a statistically binding commitment scheme and will denote the commitment algorithm as Comm (see e. g. [Gol07] for a formal definition). Throughout this paper, we omit the modulus operation $\bmod (n+1)$ to improve readability. Logarithms are always taken with base 2, i. e. $\log(x) := \log_2(x)$.

2.1 MPC-in-the-head Paradigm

Secure multiparty computation (MPC) is a very useful paradigm used in many cryptographic protocols. A MPC protocol Π_f for a function f with arity $n+1$ is a protocol played by $n+1$ parties $P_0, \ldots, P_n$. Party P_i has some secret x_i and the goal of the protocol is to compute $f(x_0, \ldots, x_n)$ such that P_i does not learn anything about x_j for $j \neq i$. The *view* $\mathrm{View}(P_i)$ of a party P_i is a string containing the secret x_i, the randomness r_i used by P_i and all messages sent to P_i and all message sent by P_i. Two views $\mathrm{View}(P_i), \mathrm{View}(P_j)$ are *consistent* if the messages sent from P_i to P_j are exactly the messages received by P_j from P_i and vice versa. Such a protocol Π_f is *perfectly correct*, if the output $\Pi_f(x_0, \ldots, x_n)$ on inputs $x_0, \ldots, x_n$ generated by the protocol equals $f(x_0, \ldots, x_n)$ for all possible randomness used by the parties. Furthermore, a protocol Π_f is *t-private* if for all $T \subseteq [0, n]$ with $|T| \leq t$, all $x_0, \ldots, x_n$, there is a simulator S that on input $(T, (x_i)_{i \in T}, f(x_0, \ldots, x_n))$ outputs a vector $(v_i)_{i \in T}$ such that $(v_i)_{i \in T}$ and $(\mathrm{View}(P_i))_{i \in T}$ have the same distribution.

Corrupted parties aiming to break the privacy of such a protocol can either be semi-honest (they follow the rules of the protocol, but try to obtain additional information from the transcripts) or malicious (they do not need to follow the protocol at all). Surprisingly, Ishai et al. [IKOS09] showed that security against semi-honest attackers is sufficient to obtain zero-knowledge proofs.

The main idea is to use the function $f_y(x) := R(x, y)$ for an instance y and split the witness x into shares $x_0, \ldots, x_n$ with $x = \bigoplus x_i$. The prover then simulates the MPC protocol for $n+1$ parties, where party i has input x_i. For each party i, this simulations generates a view and the prover uses a commitment scheme to commit to these views and sends all of the commitments to the verifier. The verifier now requests a random subset of the views from the prover, checks whether they belong to the commitments and whether they are consistent. One important advantage of this approach is the small communication complexity dominated by the size of the views and their commitments. Figure 1 contains a more formal description of this protocol.

This transformation (called the MPC-in-the-head paradigm) has been used for many applications, including theoretical ones such as the black-box construction of non-malleable commitments [GLOV12] or zero-knowledge protocols having communication complexity proportional only to the square-root of the

The verifier and the prover have input $y \in L_R$. The prover knows x such that $R(y, x) = 1$. A perfectly correct and t-private $n+1$-party MPC protocol Π_{f_y} is given $(2 \leq t \leq n)$

Commit The prover does the following:
1. Sample random vectors $x_1, \dots, x_n$. Set $x_0 = x \ominus \bigoplus_{i=1}^{n} x_i$.
2. Run $\Pi_{f_y}(x_0, \cdots, x_n)$ and obtain the views $w_i = \text{View}(P_i)$ for all $i \in [0, n]$.
3. Compute commitment $a = \mathsf{Comm}(w_0, \dots, w_n)$ and send a to the verifier.

Prove The verifier chooses a subset $E \subset [0, n]$ such that $|E| = t$ and sends it to the prover. The prover reveals the value w_e for all $e \in E$.

Verify The verifier runs the following checks:
1. Simulate the run of all parties $e \in E$ using w_e. If any output y_e is $\neq 1$, output `reject`.
2. If $\exists\{i, j\} \subset E$ such that w_i is not consistent with w_j, output `reject`.
3. Output `accept`.

Fig. 1: The MPC-in-the-head zero-knowledge protocol

verification circuit [AHIV17], but also for practically usable implementations including zero-knowledge proofs [GMO16] and signature schemes for post-quantum cryptography [CDG$^+$17].

2.2 Zero-knowledge Proofs

A *zero-knowledge proof* between a *prover* $\mathcal{P}$ and a *verifier* $\mathcal{V}$ is a two-player game. The goal of the the prover $\mathcal{P}$ is to convince the verifier $\mathcal{V}$ that they know a certain secret x without revealing *any* information about this secret. Zero-knowledge proofs are extremely useful for different cryptographic applications such as signature schemes or multi-party computations. In this work, we only need a certain kind of well-structured protocol, called a Σ-protocol. In the following, let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP-relation, i. e. for all $x, y \in \{0,1\}^*$, the value $R(x, y)$ can be computed in polynomial time and if $R(x, y) = 1$, we have $|x| \leq |y|^{O(1)}$. Here, we identify the relation R with its binary characteristic function $R\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ with $R(x, y) = 1$ iff $(x, y) \in R$. The value x is a *witness* to y. By L_R, we denote the language associated with R, i. e. $L_R = \{y \mid \exists x \text{ s.t. } R(x, y) = 1\}$. In some parts of this work, we make the relation R more explicit using a function $\phi\colon \{0,1\}^* \to \{0,1\}^*$ with the natural interpretation of $L_\phi = \{y \mid \exists x \text{ s.t. } \phi(x) = y\}$.

Definition 1 (Σ-Protocol). *The goal of the protocol $\Pi_R(y)$ between two players $\mathcal{P}$ and $\mathcal{V}$ is to convince $\mathcal{V}$ that $y \in L_R$, where $y \in \{0,1\}^*$ is known to both player. Such a protocol is called a Σ-Protocol for the relation R if it satisfies the following conditions:*

- *Π_R has the following communication pattern:*
  1. ***Commit:** $\mathcal{P}$ sends a first message **a** to $\mathcal{V}$,*
  2. ***Challenge:** $\mathcal{V}$ sends a random element **e** to $\mathcal{P}$,*
  3. ***Prove:** $\mathcal{P}$ replies with a second message **z**.*

- ***Completeness:*** *If both players $\mathcal{P}$ and $\mathcal{V}$ are honest and $\boldsymbol{y} \in L_R$, then $Pr[(\mathcal{P}, \mathcal{V})(\boldsymbol{y}) = \mathtt{accept}] = 1$.*
- *s-**Special Soundness**: For any $\boldsymbol{y}$ and any set of $s \geq 2$ accepting conversations $\{(\boldsymbol{a}, \boldsymbol{e}_i, \boldsymbol{z}_i)\}_{i\in[s]}$ with $\boldsymbol{e}_i \neq \boldsymbol{e}_j$ if $i \neq j$, a witness $\boldsymbol{x}$ for $\boldsymbol{y}$ can be efficiently computed.*
- ***Special honest-verifier ZK:*** *There exists a PPT simulator S such that on input $\boldsymbol{y} \in L_R$ and $\boldsymbol{e}$ outputs a triple $(\boldsymbol{a}', \boldsymbol{e}, \boldsymbol{z}')$ with same probability distribution as a real conversations $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$ of the protocol.*

While Σ-protocols are not zero-knowledge protocols, they can be transformed into full-fledged zero-knowledge protocols by the addition of two rounds [HL10]. Furthermore, a Σ-protocol is a *public-coin* protocol, as the verifier $\mathcal{V}$ only sends random messages. Hence, the Fiat-Shamir transformation [FS86] or the Unruh transformation [Unr15] can be used to make them non-interactive in the random oracle model. Note that the Unruh transformation always gives security against quantum adversaries, while the Fiat-Shamir transformation does not do this in general [ARU14]. Nevertheless, recently it was shown that the Fiat-Shamir transformation is still secure against quantum adversaries for a large class of protocols [Cha19,DFMS19].

2.3 ZKBoo

An important Σ-protocol based on the MPC-in-the-head paradigm is called ZKBoo [GMO16]. The goal of the protocol is to convince the verifier that the prover has an input x to an arithmetic circuit ϕ such that $\phi(x) = y$, where ϕ and y are publicly known. The general idea behind ZKBoo is the partition of ϕ into a $(2, 3)$-*decomposition*, i. e. the computation of this circuit is split into three *branches* ϕ_0, ϕ_1, ϕ_2. The input x to ϕ is furthermore split into three shares x_0, x_1, x_2 such that the computation of ϕ_i only needs the shares x_i and x_{i+1}. After this computation by the prover, the verifier chooses a random index $i \in_R \{0, 1, 2\}$ and is given the computations of ϕ_i and ϕ_{i+1} along with the inputs x_i and x_{i+1}. This information can be used to verify the computations on these branches without revealing the complete input x to the verifier. Due to the small size of the communication — roughly dominated by the number of multiplication gates in ϕ — the ZKBoo-protocol has seen wide use. We provide a detailed description of the protocol in Figure 2. The protocol has since been further optimized, e.g. by using a higher number of parties and using a prepoccessing phase [KKW18] or a further optimization of the underlying symmetric primitives [KZ20]. Most famously, an optimized version called ZKB++ is the basis of the post-quantum secure Picnic signature scheme — already in round two of the NIST standardization process [CDG+17].

2.4 Strong Non-Interference

A widely adopted defense mechanism against physical attacks on cryptographic implementations such as side channel attacks (SCA) or other methods that allow to deduce the value of some variables, is masking [CJRR99b]. Ishai et al. [ISW03] introduced the notions of *probing security* that allows to construct and formally analyze building blocks (or *gadgets*), and introduced a scalable multiplication gadget. Probing security was further refined to threshold non-interference (NI) in [BBD$^+$16]. Informally, a gadget is *t-non-interfering* (or t-probing secure) if the knowledge of t_1 intermediate variables and t_2 output variables with $t_1+t_2 \leq t$ can only lead to the leak of at most t input variables. Intuitively, a t-non-interfering gadget in combination with a mask order of $n \geq t+1$ is robust against attacks where the attacker can gain knowledge of t variables. Unfortunately, this notion of non-interference is not composable, i. e. the composition of two t-non-interfering gadgets is not necessarily t-non-interfering [CPRR13]. A more composable notion of non-interference, called *strong non-interference* (SNI) was thus proposed in [BBD$^+$16]. Informally, a gadget is t-SNI if the knowledge of t_1 intermediate variables and t_2 output variables with $t_1 + t_2 \leq t$ only allows the leak of t_1 (not t) input variables. Hence, the adversary could just have probed the t_1 input variables. More formally, this notion is defined as follows.

Definition 2 (t-SNI Security). *Let G be a gadget which takes as input $n+1$ shares $(x_i)_{0\leq i\leq n}$ and as outputs $n+1$ shares $(y_i)_{0\leq i\leq n}$. The gadget G is said to be t-SNI secure if for any set of t probed intermediate variables and any subset $\mathcal{O} \subset [0,n]$ of output indices, such that $t+|\mathcal{O}| \leq n$, there exists a subset $I \subset [0,n]$ of input indices which satisfies $|I| \leq t$, such that the t intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

Here, *perfectly simulatable* means that there is a probabilistic algorithm that on input $x_{|I}$ generates t intermediate variables and $|\mathcal{O}|$ output variables with the same probability distribution as the gadget. The notion of t-SNI security is known to provide scalable protection against a broad class of side-channel attacks up to t-th order under few additional assumptions [DDF14], where the needed number of observations grows exponentially in t. It has been widely adopted e.g. for automation of applying and checking side channel resistance in software designs [BBC$^+$19] and can be viewed as a reliable and fairly efficient method to achieve a desired degree of side channel resistance. Similarly, SNI can also be used to ensure and verify protection of hardware circuits, where special care needs to be taken to account for *glitches*, either by extending the model or by carefully placing registers to interrupt unintended asynchronous propagation of signals [FGMDP$^+$18,BGI$^+$18].

Masking schemes commonly used to protect against SCA, such as the ones achieving threshold SNI security, are also MPC protocols. In the case of masking, the parties are just different parts of the same circuit (one might call it *MPC-in-the-circuit*), forcing an attacker to consider more than t parts of the circuit in parallel to infer about secret intermediate values.

3 Probing Attacks against MPC-in-the-head

In this section we analyze the MPC-in-the-head paradigm with respect to side channel attacks such as differential power analysis.

The rest of the section is dedicated to show why MPC-in-the-head protocols are vulnerable against DPA atacks and how this can be practically exploited. In order to get a better understanding of these vulnerabilities, we introduce some notations. In general, the goal of a *probing attack* (such as a DPA attack) is to reconstruct the secret input x given to some algorithm A by obtaining values used in the computation of $\mathsf{A}(x)$. We say that an algorithm A is k-private, if at least k probes are needed to reconstruct the secret x. Combining the masking technique with masking order n and modifying the circuits used in A by using n-SNI gadgets results in an algorithm A' that is $n \cdot k$-private [BBD+16]. Now, consider the case that A' is an implementation of the n-party MPC-in-the-head zero knowledge protocol as given in Figure 1. As the protocol gives out t shares to the verifier, the privacy of A' drops down by t to $n \cdot k - t$, as t input shares are now known to the attacker.

In order to illustrate that this is not only a theoretical weakness, we study the ZKBoo protocol using $(2, 3)$-circuit decomposition as defined in Appendix A. We first show that an attack using the opened views is indeed possible by using *a single probed* value. Then, we show how an adversary can construct such a probe from side channel measurements of an implementation where sensitive intermediate values leak.

3.1 Probing ZKBoo

As described in Section 2.3, the ZKBoo protocol uses a three-party decomposition of a function ϕ and a public value y, where the prover proves the knowledge of a secret value x such that $\phi(x) = y$. As summarized in Figure 2, the prover first commits views for each party. The views contain random tapes that have been used to sample random values in the circuit as well as the vector of values for each output gadget. Depending on the challenge, the prover opens a subset of these values.

From the protocol description, it is easily seen that ZKBoo reveals two out of its three shares. Note that these shares also contain two of the input shares. Hence, obtaining the missing input share x_i via a probe allows an attacker to deduce the complete secret input x to the function ϕ. In the following, we will show how such a probe can be constructed, as demonstrated by Gellersen et al. [GSE20].

3.2 Side Channel Analysis of ZKBoo

In side channel analysis, *probes* are usually just a weak leakage of a sensitive intermediate value, obtained from several measurements. For ZKBoo, we assume the same scenario as in Subsection 3.1, where two of three shares (x_0, x_1) are revealed by the protocol. As described above, even in the simplest scenario, the input views can be used to implement a side-channel attack. We further assume a leakage model where an implementation leaks weak and noisy information about each intermediate variable, separately and independently in observable measurement traces t_ℓ. The in MPC-in-the-head measurements have a weak and noisy

A (2,3)-decomposition of a function ϕ is given as Π_ϕ. The verifier and the prover have input $y \in L_\phi$. The prover knows x such that $y = \phi(x)$.

Commit: The prover does the following:

1. Generate random tapes R_0, R_1, R_2.
2. Run $\Pi_\phi(x)$ with randomness R_0, R_1, and R_2 to obtain views w_0, w_1, w_2 and outputs y_0, y_1, y_2.
3. Commit to $c_i = \mathsf{Comm}(w_i, R_i)$ for $i \in [0, 2]$.
4. Send $a = (y_0, y_1, y_2, c_0, c_1, c_2)$.

Prove: The verifier chooses an index $e \in [0, 2]$ and sends it to the prover. The prover answers to the verifier's challenge sending opening c_e, and c_{e+1} thus revealing $z = (R_j, w_j)$ for $j = \{e, e + 1\}$.

Verify: The verifier runs the following checks:

1. If $Rec(y_0, y_1, y_2) \neq y$, output `reject`.
2. If $\exists\ i \in \{e, e + 1\}$ such that $y_i \neq \text{Output}_i(w_i)$, output `reject`.
3. If $\exists\ j$ such that $w_e^j \neq \phi_e^{(j)}(w_e, R_e, w_{e+1}, R_{e+1})$, output `reject`.
4. Output `accept`.

Fig. 2: ZKBoo protocol as defined by Giacomelli et al. [GMO16].

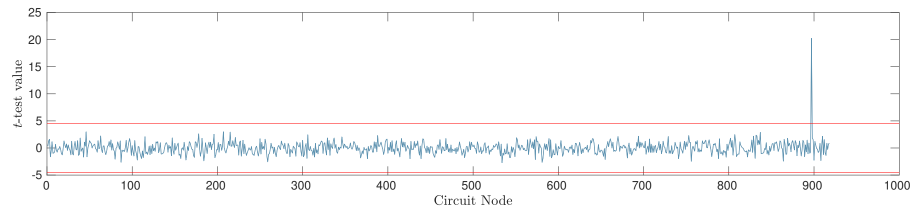


Fig. 3: A t-test based leakage analysis of a multiplication gadget in ZKBoo using the classification $\mathcal{T}_i$ where $i = w_0^j \oplus w_1^j$. The details of the experimental setup and formulation can found in Section 5.1.

dependence on x_2, which can be exploited due the revealed shares x_0 and x_1, as also shown in [GSE20]. In order to validate the straightforward exploitability, we use a simple t-test setup, where we collect a set of synthetically generated traces that corresponds to a function evaluation that uses (2,3)-circuit decomposition of ZKBoo. The analysis uses the side channel information of the unopened view for an adversarial probe $t_\mathcal{A}$ and the two opened shares of a multiplication gadget. In order the show the noisy dependence on x_2, we first target a single multiplication gadget ϕ^j that outputs three shares as $w_i^{(j+1)} = \phi_i^j(w_i^{(j)}, R_i, w_{i+1}^{(j)}, R_{i+1})$ for $i \in [0, 2]$. We classify the sets into two groups depending on the value of $w_0^{(j)} \oplus w_1^{(j)}$. The result of the analysis in Figure 3 shows the clear dependence between unrevealed share x_2 and observable measurement traces.

We consider opened views as a part of probing values in Definition 2. Thus we show that an additional probe *shatters* the independence between side channel traces and the sensitive variables.

Formally speaking, a single adversarial probe disables the simulators capability (which is defined in Definition 2) to simulate variables using a set of independent and uniformly chosen variables. Hence the multiplication gadget used in ZKBoo is not sufficient to guarantee privacy. Using the discussion given above and we can formally restate the simulation in Definition 2 as follows:

Definition 3 ($(t_{\mathcal{A}}, t_{\mathcal{E}})$-SNI Security for MPC-in-the-head protocol). *Let G be a gadget which takes as input $n+1$ shares $(x_i)_{0\leq i\leq n}$ and as outputs $n+1$ shares $(y_i)_{0\leq i\leq n}$. The gadget G is said to be t-SNI secure if for any set of $t_{\mathcal{A}}$ probed intermediate variables and $t_{\mathcal{E}}$ opened variables and any subset $\mathcal{O} \subset [0,n]$ of output indices, such that $t = t_{\mathcal{A}} + t_{\mathcal{E}}$ and $t + |\mathcal{O}| \leq n$, there exists a subset $I \subset [0,n]$ of input indices with $|I| \leq t$, such that the t intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

Definition 3 is equivalent to Definition 2 if $t_{\mathcal{E}} = 0$, if there exists no opened values. More formally, t-SNI implies (t_1, t_2)-SNI for all $t_1 + t_2 \leq t$. Nevertheless, the above definition captures the intuition that protocols following the MPC-in-the-head paradigm leak information all by themselves. Furthermore, this leaked data might be chosen carefully such that $t_{\mathcal{E}}$ leaked output variables only give information on $t_{\mathcal{E}}/2$ input variables. In such a case, using a t-SNI gadget might actually give a $(t_{\mathcal{A}}, t_{\mathcal{E}})$ gadget with $t_{\mathcal{A}} + t_{\mathcal{E}}/2 = t$ giving a more fine-granular view. Using this notion, we can design MPC-in-the-head protocols that achieve t-SNI security even if a subset of the views are revealed. In the next section we provide a circuit decomposition and define our protocol.

4 Constructing SNI-secure Decompositions

In this Section we introduce a decomposition of an arithmetic circuit secure in the SNI notion. We start with a generic decomposition definition that will be used for the circuit decomposition in the following sections. In [GMO16], the notion of a $(2,3)$-decomposition was introduced. Informally, such a decomposition splits a function ϕ into three *branches* such that the computations of two of those branches are not enough to reconstruct the complete computation of the function. A formal definition is given in Appendix A. In ZKBoo, two of these branches are revealed, while the third branch stays hidden. As shown in Section 3.1, this allows power attacks against ZKBoo, as a single probed value from this third branch might be sufficient to reconstruct the complete computation. In this section, we thus aim to construct function decompositions that are SNI-secure and withstand such probes. We first introduce a generalization of $(2,3)$-decompositions, called $(k, n+1)$-decompositions. Defining *balanced* SNI-secure versions of the multiplication gadget and the refresh gadget allows us to construct a $(\lceil n/2 \rceil + 1, n+1)$-decomposition for functions represented by arithmetic circuits. Moreover, we show that this decomposition is n-SNI secure.

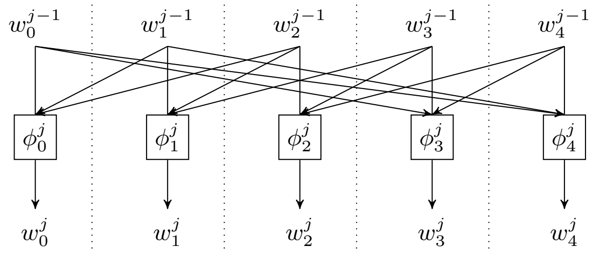


Fig. 4: The representation of the branches for the j-th gadget ϕ^j of the ($k = 3, n = 4$) decomposition for the function ϕ. Observe that each branch requires at most $k = 3$ views.

4.1 Decomposing a Function

Let $\phi\colon X \to Y$ be an arbitrary function. The protocol is performed on an input value $x \in X$ that computes $\phi(x) = y$. We use a transformation on the function ϕ to split the evaluation and the data into $n+1$ branches such that revealing n of them brings no information about the secret value x. The first step is to apply a surjective (possibly randomized) algorithm Share to x to split it into input shares $x_0, \dots, x_n$. The input shares and the intermediate values for the i-th branch is stored in w_i, which is called a *view*, and contains $(d+1)$ elements $w_i^{(0)}, \dots, w_i^{(d)}$. The single steps of the computation are described by a set of $(n+1)\cdot(d+1)$ functions $\mathcal{F} = \{\phi_i^{(j)} \mid \forall\ 0 \leq i \leq n \text{ and } 0 \leq j \leq d\}$. In order to guarantee that k views are sufficient to recompute a single branch, the functions ϕ_i^j take input from the k branches $i, i+1, \dots, i+(k-1)$. The 0-th value $w_i^{(0)}$ of a view w_i is simply its input share x_i. The remaining values $w_i^{(j)}$ can be computed in the following iterative way:

$$w_i^j = \phi_i^j((w_m^{[0,j-1]}, R_m))_{i \leq m \leq i+(k-1)} \text{ for } 0 \leq i \leq n,$$

where $w_m^{[0,j-1]} = (w_m^{(0)}, \dots, w_m^{(j-1)})$. Here, R_i denotes the source of randomness within the i-th branch. As an example we can see the visual representation of ϕ^j for $n+1 = 5$ in Figure 4.

After evaluating the d gadgets, the output value y_i is computed from w_i by the functions Output_i, i.e. $y_i = \mathsf{Output}_i(w_i)$. Finally, the output values y_i are recombined as $\mathsf{Rec}(y_0, \dots, y_n) = y = \phi(x)$.

Finally we can introduce the complete $(k, n+1)$-decomposition definition generalizing the definition given in [GMO16].

Definition 4. *A $(k, n+1)$-decomposition $\mathcal{D}$ of a function $\phi\colon X \to Y$ is a set of functions*

$$\mathcal{D} = \{\mathit{Share}, (\mathit{Output}_i)_{0 \leq i \leq n}, \mathit{Rec}\} \cup \mathcal{F},$$

such that Share, Output_i, Rec, and $\mathcal{F}$ are defined as above. Let Π_ϕ be the evaluation protocol defined in Fig. 5.

The decomposition must also have the following properties:

Let $\phi\colon X \to Y$ be a function and $\mathcal{D}$ be a $(k, n+1)$-decomposition of ϕ. For an input $x \in X$, perform the following:

1. Generate the random tapes R_i for $0 \leq i \leq n$.
2. Generate the secret shares: $(x_0, \ldots, x_n) \leftarrow \mathsf{Share}(x; r_1, \ldots, r_n)$ where r_i is sampled from the random tape R_i.
   – Initialise $w_i^{(0)} \leftarrow x_i$ for $0 \leq i \leq n$.
   – for $1 \leq j \leq d$ compute

$$w_i^{(j)} = \phi_i^{(j)}((w_m^{[0,j-1]}, R_m))_{i \leq m \leq i+(k-1)} \text{ for } 0 \leq i \leq n$$

.

3. Compute $y_i = \mathsf{Output}_i(w_i, R_i)$ for $0 \leq i \leq n$.
4. Output $y = \mathsf{Rec}(y_0, \ldots, y_n)$.

Fig. 5: A protocol Π_ϕ using a decomposition $\mathcal{D}$ to evaluate $\phi(x)$. The figure is adapted from [GMO16].

– ***Correctness:*** *$Pr[\phi(x) = \Pi_\phi(x)] = 1$ for all $x \in X$ where the probability is over the random choices.*
– n-***Privacy:*** *The protocol is correct and for all $e \in [0, n]$ there exists a PPT algorithm S_e such that the distribution $S_e(\phi, y)$ and the distribution $(\{R_i, w_i\}_{i \in \{e, e+1, \ldots, e+(n-1)\}}, y_{e-1})$ are statistically indistinguishable.*

The goal of the next subsection is the construction of an $(\lceil n/2 \rceil + 1, n+1)$-decomposition for functions $\phi\colon \mathbb{K}^k \to \mathbb{K}^\ell$ implemented by an arithmetic circuit. Furthermore, we want this decomposition to be n-SNI to prevent the attacks described in Section 3.1. Note that the construction of a $(n, n+1)$-decomposition is just a simple generalization of the linear $(2, 3)$-decomposition of [GMO16]. The main technical problem to construct an $(\lceil n/2 \rceil + 1, n+1)$-decomposition is the fact that each gate/function $\phi_i^{(j)}$ can have inputs only from branches $i, i+1, \ldots, i + \lceil n/2 \rceil$. Taking a closer look at the existing construction of gadgets against side-channel attacks for multiplication (for example, the ISW gadget of [ISW03] or the more refined version of [RP10]) shows that the computation of the i-th branch depends on i other branches. To guarantee that each branch depends only on $\lceil n/2 \rceil$ other branches, we construct *balanced* gadgets.

4.2 Constructing Balanced Gadgets

Next we focus on the gadgets. As gates such as unary addition, unary multiplication, and binary addition are linear, there is no need for secure gadgets for these operations. We thus only need to examine the two essential SNI -secure gadgets needed for the multiplication operation. To obtain a secure multiplication operation, a *refresh gadget* is also needed, whenever a variable is used in multiple multiplication gates. See e. g. [BBC$^+$19] for a more formal treatment.

We need to analyze and adapt these gadgets because all known SNI-secure gadgets have an unbalanced structure, which causes the need for more than

$\lceil n/2 \rceil$ other views to compute some output share. Therefore, the main purpose is to generate gadgets such that every branch needs at most $\lceil n/2 \rceil$ other input shares in order to compute the corresponding output share.

Balancing the Multiplication Gadget First, we shortly remark the multiplication gadget defined in [RP10] and proven to be secure in t-SNI in [BBD$^+$15a]. Let $(x_0, \ldots, x_n)$ and $(y_0, \ldots, y_n)$ be the shares of the two sensitive variable x and y. The multiplication gadget to calculate the output shares $(z_0, \ldots, z_n)$ of $z = xy$ can be summarized in two steps as follows:

1. Calculate $r_{j,i} = (r_{i,j} \oplus x_i y_j) \oplus x_j y_i$ where $r_{i,j} \in_R \mathbb{K}$ for $0 \leq i < j \leq n$.
2. Calculate $z_i = x_i y_i \oplus \bigoplus_{j=0; j \neq i}^{n} r_{i,j}$ for $0 \leq i \leq n$.

As seen in the description above, the calculation of z_i requires $n - i$ random values ($r_{i,j}$ such that $i < j$) and i intermediate products ($r_{i,j}$ such that $i > j$). In order to generate a $(\lceil n/2 \rceil + 1, n)$-decomposition, we need to have a *balanced* multiplication gadget such that every index requires the same number of random values and intermediate products.

Informally, we can illustrate the intermediate values of the multiplication gadget as a matrix $\mathtt{A}$ as follows:

$$\mathtt{A}_{i,j} = \begin{cases} r_{i,j} \in_R \mathbb{K}, & \text{for } i < j \\ (r_{i,j} \oplus x_i y_j) \oplus x_j y_i, & \text{for } i > j \\ x_i y_i, & \text{for } i = j \end{cases}$$

Hence we can represent output shares as $z_i = \bigoplus_{j=0}^{n} \mathtt{A}_{i,j}$. Using this representation, $n(n+1)/2$ random values (and intermediate products) can be reorganised in such a way that each row contains exactly $\lceil n/2 \rceil$ of them. In order to do so, we define for $i \in [0, n]$ the intervall $J_i = [i - \lceil n/2 \rceil, i + \lceil n/2 \rceil] \setminus \{i\}$, where *no modular arithmetic is used*, e. g. $J_0 = [1, \lceil n/2 \rceil]$ and $J_n = [n - \lceil n/2 \rceil, n - 1]$. In order to generate a *balanced* multiplication gadget, one can take a partial transpose of the matrix $\mathtt{A}$ as follows:

$$\mathtt{A}'_{i,j} = \begin{cases} x_i y_i, & \text{for } i = j \\ r_{i,j} \in_R \mathbb{K}, & \text{for } i \neq j, j \notin J_i \\ (r_{i,j} \oplus x_i y_j) \oplus x_j y_i, & \text{for } j \in J_i \end{cases}$$

As an example, let us illustrate the multiplication gadget and the balanced multiplication gadget for $n + 1 = 5$:

$$\mathtt{A} = \begin{bmatrix} x_0y_0 & r_{0,1} & r_{0,2} & r_{0,3} & r_{0,4} \\ r_{1,0} & x_1y_1 & r_{1,2} & r_{1,3} & r_{1,4} \\ r_{2,0} & r_{2,1} & x_2y_2 & r_{2,3} & r_{2,4} \\ r_{3,0} & r_{3,1} & r_{3,2} & x_3y_3 & r_{3,4} \\ r_{4,0} & r_{4,1} & r_{4,2} & r_{4,3} & x_4y_4 \end{bmatrix} \quad \mathtt{A}' = \begin{bmatrix} x_0y_0 & r_{1,0} & r_{2,0} & r_{0,3} & r_{0,4} \\ r_{0,1} & x_1y_1 & r_{2,1} & r_{3,1} & r_{1,4} \\ r_{0,2} & r_{1,2} & x_2y_2 & r_{3,2} & r_{4,2} \\ r_{3,0} & r_{1,3} & r_{2,3} & x_3y_3 & r_{4,3} \\ r_{4,0} & r_{4,1} & r_{2,4} & r_{3,4} & x_4y_4 \end{bmatrix}$$

The matrix $\mathtt{A}$ represents the multiplication gadget defined in [RP10] and matrix $\mathtt{A}'$ describes the equivalent, but balanced multiplication gadget. The parts transposed can be seen in grey. We can see that in both cases $z_i = \bigoplus_{j=0}^{n} \mathtt{A}_{i,j}$ and $z'_i = \bigoplus_{j=0}^{n} \mathtt{A}'_{i,j}$. Although the shares are calculated differently i. e. $z_i \neq z'_i$, the correctness of the gadgets holds i. e. $z = xy = \bigoplus_{i=0}^{n} z_i = \bigoplus_{i=0}^{n} z'_i$.

Finally, we formally introduce the *balanced* multiplication gadget to calculate the output shares $(z_0, \ldots, z_n)$ of $z = xy$ as follows:

1. Calculate $r_{j,i} = (r_{i,j} \oplus x_i y_j) \oplus x_j y_i$ where $r_{i,j} \in_R \mathbb{K}$ for $0 \leq i < j \leq n$.
2. Calculate $z_i = x_i y_i \oplus \bigoplus_{j=0;j\neq i}^{n} \delta_{i,j}$ for $0 \leq i \leq n$ where $\delta_{i,j}$ defined as:

$$\delta_{i,j} = \begin{cases} r_{i,j}, & \text{for } j \in J_i \\ r_{j,i} & \text{else .} \end{cases} \tag{1}$$

Remark that the balanced multiplication gadget defined above does not bring any overhead to the scheme. The explicit description can be found in Appendix B, Algorithm 2. In the final step, we show that the balanced multiplication gadget indeed satisfies the SNI notion, as the gadget is secure against n attack probes.

Theorem 1 (n-SNI Security for balanced multiplication gadget). *Let G be the balanced multiplication gadget which takes $(x_i)_{0\leq i\leq n}$ and $(y_i)_{0\leq i\leq n}$ as the input shares, and outputs $(z_i)_{0\leq i<n}$. For any set of $t \leq n$ intermediate variables and any subset $\mathcal{O} \subset [z_0, \ldots, z_n]$ of output shares such that $t + |\mathcal{O}| \leq n$, there exists a subset $I \subset [0, n]$ of input indices which satisfies $|I| \leq t$, such that the t intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

Proof. In order to prove the theorem, we use a similar structure as in [BBD+15a] and show that every set of intermediate variables with t elements can be simulated by two sets of input shares $(x_i)_{i\in I}$ such that $|I| \leq t$ and $(y_j)_{j\in J}$ such that $|J| \leq t$. We divide the probes in four groups:

A1: If x_i, y_i, or $x_i y_i$ is probed, add i to I and J.
A2: If $r_{i,j}$ add i to I and J; If $z_{i,j}$ is probed we need to consider two cases:
– If $j > \lceil n/2 \rceil$ add i to I and J.
– If $j \leq \lceil n/2 \rceil$ add j to I and J.

Let U be the common values added to I and J by **A1** and **A2**.

A3: If $x_i y_j \oplus r_{i,j}$ is probed and if $i \in U$ or $j \in U$ add i and j to both I and j.
A4: If $x_i y_j$ is probed, add i to I and j to J.

Clearly, $|I|$ and $|J|$ have at most one index per probe, and therefore $|I| \leq t$ and $|J| \leq t$. Remark that for $i < j$, the variable $r_{i,j}$ used in $z_{i,k}$ — the k-th partial sum of z_i — if $j > \lceil n/2 \rceil$, otherwise it is used in the partial sum $z_{j,k}$. The variables in the class of **A1** and **A2** can be simulated clearly according to our selection. Let us consider the following cases for $i < j$:

– If $\{i, j\} \in U$, according to our selection we can simulate every variables $r_{i,j}$, $x_i y_j$, $x_i y_j \oplus r_{i,j}$, $x_j y_i$ and $r_{j,i}$.
– If $i \in U$ and $j \notin U$, then we can simulate $r_{i,j}$ as in the circuit (by sampling a random value) and compute $x_i y_j \oplus r_{i,j}$ since $i \in U$ and $j \in J$ according to our selection.
– If $i \notin U$ and $j \in U$, this means that $r_{i,j}$ is not probed and not used any probed value (since $i \notin U$). Therefore, we can assign $r_{j,i}$ as a random value. If $x_i y_j \oplus r_{i,j}$ is also probed we can simply simulate this value as $x_i y_j \oplus r_{i,j} = x_j y_i \oplus r_{j,i}$ since $j \in U$ and $i \in J$
– If $i \notin U$ and $j \notin U$ and if $x_i y_j \oplus r_{i,j}$ is probed, since $r_{i,j}$ is not probed and does not enter into the computation of any other probed variable, we can perfectly simulate such probe with a random value.

From the above discussion we can see that, we can perfectly simulate any variable $r_{i,j}$ such that $i \in U$. Observe that the only difference of our gadget is in class of **A2** where some $r_{i,j}$ is used either in the partial product of $z_{i,k}$ or $z_{j,k}$. However in both cases we cover the required indices according to our selection. Therefore we can perfectly simulate all partial sums $z_{i,k}$. Hence all probed variables are perfectly simulated.

In the last part of the proof we consider the simulation of the subset of output shares $z_{|\mathcal{O}}$ from $x_{|I}$ and $y_{|J}$. From the discussion above, we can see that we can simulate outputs z_i with $i \in U$ perfectly. Now, consider z_i with $i \notin U$. We construct a set of indices V as follows: For each variable $x_i y_j \oplus r_{i,j}$ in **A3** with $i \notin U$ and $j \notin U$ (the fourth case described above), we add $j \in V$ if $i \in \mathcal{O}$ or $i \in V$ if $i \notin \mathcal{O}$. Note that we only considered variables in **A3**, where we increased I and J by two elements. As V was only increased by one element, we have $|U| + |V| \leq t$ and thus $|U| + |V| + |\mathcal{O}| \leq n$. Hence, there is an index $j^* \in [0, n]$ such that $j^* \notin (U \cup V \cup \mathcal{O})$. By definition, we have

$$z_i = x_i y_i \oplus \bigoplus_{j=0; j \neq i}^{n} \delta_{i,j} = \delta_{i,j^*} \oplus \left(x_i y_i \oplus \bigoplus_{j=0; j \neq i; j \neq j^*}^{n} \delta_{i,j} \right).$$

We will now show that r_{i,j^*} and $r_{j^*,i}$ (the only possible values for δ_{i,j^*}) are not used in the computation of any probed intermediate variable or another output variable $z_{i'}$ with $i' \in \mathcal{O}$. As $i \notin U$, neither r_{i,j^*} nor any partial sum $z_{i,k}$ was probed. Furthermore, as $j^* \notin U$, neither $r_{j^*,i}$ nor any partial sum $z_{j^*,k}$ were probed. Finally, $j^* \notin \mathcal{O}$ and z_{j^*} was also not probed. Hence, r_{i,j^*} and $r_{j^*,i}$ are not used in the computation of a probed intermediate variable. We still need to show that is not needed for other output variables $z_{i'}$. If $i < j^*$, then $x_i y_{j'} \oplus r_{i,j^*}$ was not probed, as $j^* \notin V$ and $i \in \mathcal{O}$. If $j^* < i$, then $x_{j*} y_i \oplus r_{j^*,i}$ was note probed, as $j^* \notin (V \cup \mathcal{O})$. Hence, r_{i,j^*} and $r_{j^*,i}$ are not used in the computation of any output variable $z_{i'}$ and we simulate z_i by sampling a random value. □

Balancing the RefreshMask Gadget In the next section we focus on balancing another important gadget for SNI notion: the RefreshMask gadget. The foundation of our gadget relies on the gadget defined in [BBD+15a], can be found in Appendix B, Algorithm 3. Remark that this gadget is an essential part of the SNI notion, due to its role in composability. Informally speaking, refresh masking gadgets are used to protect circuits where a set of inputs $(x_0, \ldots, x_n)$ is used in more then one multiplication gadget. An example of such a circuit can be found in [RP10] for the function $\phi(x) = x^{254}$. Thus, the usage of RefreshMask depends on the structure of the underlying circuit.

Clearly the total number of required randomness in Algorithm 3 is $n(n+1)/2$. Moreover as seen in Line 5 of the algorithm, x'_i requires $n-i$ random values. Using a similar strategy as in Section 4.2, we can reformulate this gadget and generate a balanced gadget, where each index requires the same number of randomness.

Algorithm 1 Balanced RefreshMask Gadget

```
Input: The vector of shares $(x_0, \ldots, x_n)$.
Output: The vector of shares of $x$ as $(x'_0, \ldots, x'_n)$.
1: for $0 \leq i \leq n$ do
2:     $x'_i \leftarrow x_i$
3: for $0 \leq i \leq n$ do
4:     for $0 \leq j \leq \lceil n/2 \rceil$ do
5:         $r_{i,i+j} \leftarrow rand()$                      ▷ $r_{i,i+j} \in_R \mathbb{K}$
6:         $x'_i \leftarrow x'_i \oplus r_{i,i+j}$             ▷ Denoted by $a_{i,i+j}$
7:         $x'_{i+j} \leftarrow x'_{i+j} \oplus r_{i,i+j}$     ▷ Denoted by $b_{i+j,i}$
8: return $(x'_0, \ldots, x'_n)$
```

Theorem 2 (n-SNI Security for Balanced RefreshMask gadget). *Let G be the balanced RefreshMask gadget that used in the MPC-in-the-head protocol which takes $(x_i)_{0\leq i\leq n}$ and outputs $(x'_i)_{0\leq i<n}$. For any set of $t \leq n$ intermediate variables and any subset $\mathcal{O} \subset [0, n]$ of output shares such that $t + |\mathcal{O}| \leq n$, there exists a subset $I \subset [0, n]$ of input indices which satisfies $|I| \leq t$, such that the t intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

Proof of the Theorem 2 can be found in Appendix B. After introducing suitable gadgets where each branch needs at most $\lceil n/2 \rceil$ values from other branches, we can finally introduce the complete circuit decomposition.

4.3 A $(\lceil n/2 \rceil + 1, n + 1)$-Decomposition for Arithmetic Circuits

Let $\phi \colon \mathbb{K}^k \to \mathbb{K}^\ell$ be a function implementable by an arithmetic circuit with d gates. The branches for $n+1$ shares are initialised by the Share algorithm that on input $x \in \mathbb{K}^k$ and random values $r_1, \ldots, r_n$ produces the input shares $x_0, \ldots, x_n$ with $x_i = r_i$ for $i = 1, \ldots, n$ and $x_0 = \bigoplus_{i=1}^n x_i \ominus x$. The reconstruction function $\mathsf{Rec}(y_0, \ldots, y_n)$ is defined as $\mathsf{Rec}(y_0, \ldots, y_n) = \bigoplus_{i=0}^n y_i$.

Depending on the gates used in the arithmetic circuit, we can define the set of functions $\mathcal{F} = \{\phi_i^{(j)} \mid \forall\ i \in [0, n] \text{ and } j \in [0, d]\}$ as follows:

- If the j-th gate corresponds to an affine function: $ax \oplus b$ where $a, b \in \mathbb{K}$,

$$\phi_i^{(j)} = \begin{cases} ax_i \oplus b, & \text{for } i = 0 \\ ax_i & \text{else .} \end{cases}$$

- If the j-th gate corresponds to the addition of two sensitive variables: $x \oplus y$

$$\phi_i^{(j)} = x_i \oplus y_i.$$

- If the j-th gate is a multiplication of two sensitive variables: xy

$$\phi_i^{(j)} = x_i y_i \oplus \bigoplus_{i=0}^{n} \delta_{i,j} \text{ for } i \neq j \text{ and } \delta_{i,j} \text{ as in Equation (1)}$$

  Remark that the random values $r_{i,j}$ used in $\delta_{i,j}$ are sampled from R_i.
  If a variable x_i is *not* used for the first time in such a multiplication, we replace x_i by

$$x_i \oplus \bigoplus_{j=1}^{\lceil n/2 \rceil} r_{i,i+j} \oplus \bigoplus_{j=1}^{\lceil n/2 \rceil} r_{i-j,i}$$

  where $r_{i,j}$ is chosen as in Algorithm 1, i. e. we first apply the balanced Refresh gadget.

Finally we can define the output as $\mathsf{Output}(w_i, R_i) = w_i^{(d)}$.

Proposition 1. *The decomposition*

$$\mathcal{D} = \{\mathit{Share}, (\mathit{Output}_i)_{0 \leq i \leq n}, \mathit{Rec}\} \cup \mathcal{F}$$

as defined above is an $(\lceil n/2 \rceil + 1, n + 1)$*-decomposition.*

Proof. We start with the correctness of our protocol.

The correctness of the decomposition follows from the masking structure. Remark that the decomposition is based on well-known masking techniques and secure gadgets which are known to be functionality preserving. Since any combination of our gadgets is also correct $Pr[\phi(x) = \Pi_\phi(x)] = 1$ for all choices or randomness.

In the second part of the proof we define the simulator S_e for an index e and on inputs ϕ and y. For the sake of simplicity we define the set of indices $[e, e + n - 1]$ as E and denote the last remaining index as $\tilde{e} = e - 1$.

- Sample the random tapes $(\tilde{R}_i)_{i \in E}$
- Initialise $\tilde{w}_i^{(0)}$ by sampling a random value from $\tilde{R}_i$ for $i \in E$. Then for all linear gadgets (addition and affine) calculate the values using the corresponding functions $\phi_i^{(j)}$ for all $i \in E$. If the gadget is a multiplication gadget, we do the following:

  - For all computations $\phi_i^{(j)}$ that require the view $w_{\tilde{e}}$, we randomly sample $w_i^{(j)}$.
  - For all other views, we simply compute $\phi_i^{(j)}$, since the simulation already has the knowledge of the required views.
- Calculate $\tilde{y}_i = \mathsf{Output}(\tilde{w}_i, \tilde{R}_i)$ for all $i \in E$.
- Calculate $\tilde{y}_{\tilde{e}} = y \ominus \bigoplus_{i \in E} \tilde{y}_i$
- Output $\mathcal{O} = ((\tilde{w}_i, \tilde{R}_i)_{i \in E}, \tilde{y}_{\tilde{e}})$

We can see that $\mathcal{O}$ that is outputted by S_e has the same distribution as the real values $((w_i, R_i)_{i \in E}, y_{\tilde{e}})$ provided by Π_ϕ. Observe that all the elements of S_e are calculated as the same functions in the protocol except for the second case of the multiplication gadget. In this case randomly sampling the required values $w_i^{(j)}$ is a valid approach since $w_{\tilde{e}}$ contains a random value sampled from $R_{\tilde{e}}$, which is uniformly random. We can conclude that $\mathcal{D}$ has n-privacy. □

Proposition 2. *Let $\mathcal{D}$ be the $(\lceil n/2 \rceil + 1, n+1)$-decomposition described above. Let Π_ϕ be the protocol described in Fig. 5. Then Π_ϕ is n-SNI. The length of each view of Π_ϕ is $(k + N_\otimes + \ell) log |\mathbb{K}| + \kappa$, where $N_\otimes$ is the number of multiplication gates in the arithmetic circuit implementing ϕ.*

Proof. All linear gadgets are n-SNI by definition, Theorem 1 shows that our balanced multiplication gadget is n-SNI, and Theorem 2 shows that our balanced refresh gadget is n-SNI. As each input to a multiplication gadget is used at most once (due to the RefreshMask gadget), Lemma 3 in [BBD+15b] implies that Π'_ϕ is n-SNI.

Finally, we need to analyze the ingredients of a view in order to reveal the size of it. By definition, the function ϕ takes k inputs and produces ℓ outputs where each value can be represented by $\log(|\mathbb{K}|)$ bits. Moreover the views need a security parameter κ to generate random tapes. Note that the computation between two multiplication gates can be compressed as in [GMO16]. Hence, the size of w_i can be calculated as:

$$(k + N_\otimes + \ell) \log(|\mathbb{K}|) + \kappa$$

□

In the next Section we provide a version of ZKBoo that can be extended to arbitrary orders and provide probing security despite of the opened views.

5 $(n+1)$-ZKBoo Protocol

In this section, we provide our zero knowledge proof based on the ZKBoo protocol [GMO16] that satisfies the SNI-security notion. The main idea is using the same structure of ZKBoo, but changing the underlying MPC-in-the-head protocol with $(\lceil n/2 \rceil + 1, n+1)$ circuit decomposition. Thus, our scheme can resist $n - \lceil n/2 \rceil - 1$ probing attacks with $\lceil n/2 \rceil + 1$ opened views.

Fig. 6: The $(n+1)$-ZKBoo protocol

A $(\lceil n/2\rceil+1, n+1)$ decomposition of function ϕ is given with N gadgets. The verifier and the prover have input $y \in L_\phi$. The prover knows x such that $y = \phi(x)$. Let Π_ϕ be the protocol given in Figure 5.

Commit: The prover does the following:
1. Generate random tapes R_i for $0 \leq i \leq n$.
2. Run $\Pi_\phi(x)$ with randomness $R_0, \ldots, R_n$ to obtain views w_i and outputs y_i for $0 \leq i \leq n$.
3. Commit to $c_i = \mathsf{Comm}(w_i, R_i)$ for $0 \leq i \leq n$.
4. Send $\mathbf{a} = (y_i, c_i)_{0 \leq i \leq n}$.

Prove: The verifier choose an index $\mathbf{e} \in [0, n]$ and sends it to the prover. The prover answers by sending opening $\mathbf{z}_e = (c_i)_{e \leq i \leq e+\lceil n/2\rceil}$ thus revealing (R_i, w_i) for $e \leq i \leq e + \lceil n/2\rceil$.

Verify: The verifier runs the following checks:
1. If $\mathsf{Rec}(y_0, y_1, \ldots, y_n) \neq y$, output `reject`.
2. If $\exists\ i \in [e, e + \lceil n/2\rceil]$ such that $y_i \neq \mathsf{Output}_i(w_i)$, output `reject`.
3. If $\exists\ j$ such that $w_i^{(j)} \neq \phi_i^{(j)}((w_k, R_k)_{e \leq k \leq e+\lceil n/2\rceil})$ for all $e \leq i \leq e + \lceil n/2\rceil$, output `reject`.
4. Output `accept`.

A brief summary of the zero-knowledge proof can be described as follows. Assume that an $(\lceil n/2\rceil + 1, n)$-decomposition for the function ϕ is given. The prover uses the private input x to run the protocol given in Figure 5 that satisfies $\phi(x) = y$, where y is a public value. After running the protocol, the prover commits views $w_0 \ldots, w_n$. In the second step, the verifier challenges the prover using an index $e \in [0, n]$ and prover opens views for all w_i with $i \in [e, e+\lceil n/2\rceil]$. Remark that each output share depends on exactly $\lceil n/2\rceil + 1$ consecutive views $w_e, \ldots, w_{e+\lceil n/2\rceil}$. Hence, opening $\lceil n/2\rceil + 1$ views is enough to calculate each output value $w_e^{(j)}$ for all $0 \leq j \leq N$. Hence, the verifier accepts if the opened views are consistent with the committed values. The summary of the protocol can be found in Figure 6.

Proposition 3. *The $(n+1)$-ZKBoo protocol given in Figure 6 with two parties $\mathcal{P}$ as prover and $\mathcal{V}$ as verifier is a Σ-protocol for the relation $\phi(x) = y$ with $n+1$-special soundness.*

Proof. Clearly, the $(n+1)$-ZKBoo protocol follows the communication pattern of a Σ-protocol. Due to the underlying gadgets' properties of the MPC-in-the-head paradigm, if both parties follow the correct communication pattern, i.e. if both parties are honest, then $Pr[(\mathcal{P}, \mathcal{V})(y) = \texttt{accept}] = 1$. Hence, the $(n+1)$-ZKBoo protocol is complete.

In order to prove the special soundness of the protocol, we need to analyze $n+1$ accepted conversations $\{(\mathbf{a}, \mathbf{e}, \mathbf{z_e})\}$ with $e = 0, \ldots, n$. Clearly, the accepted conversations reveal (R_i, w_i) for $i = 0, \ldots, n$. Thanks to the binding property of the commitment scheme, the views corresponding to the same index for different challenges are equal. That is, for two different challenges $\mathbf{z}_e$ and $\mathbf{z}_{e'}$ the views corresponding the same index are equal, i.e. $w_i \in \mathbf{z}_e$ and $w_i \in \mathbf{z}_{e'}$ are equal. Similarly, $R_i \in \mathbf{z}_e$ also equals $R_i \in \mathbf{z}_{e'}$. As all conversations are accepted, we have $y_i = \mathsf{Output}_i(w_i)$ for $i \in [0, n]$. Moreover, we know that every entry $w_i^{(j)}$ in w_i was computed correctly by the corresponding function $\phi_i^{(j)}$, as all branches were checked by the verifier. Hence, we can traverse the decomposition bottom-up to reconstruct all input shares $x_i = w_i^{(0)}$. Finally, we we can calculate $\mathsf{Rec}(x_0, \ldots, x_n) = x$ correctly. Hence, we have $\phi(x) = y$ and the $(n+1)$-ZKBoo-protocol thus has $n+1$-special soundness.

Note that to be able to correctly calculate the input x, all the branches should be checked. Assume that the number of accepted conversations is less then n. Although the challenges might contain all views, not all branches were checked by the verifier. While we are now able to check the branches ourself, if any branch contains an error, we are not able reconstruct x.

For the special honest-verifier ZK, we will now construct a simulator S working on input e and $y \in L_\phi$. Its goal is to produce a triple $(\mathbf{a}', \mathbf{e}, \mathbf{z}_e)$ with the same probability distribution as the protocol. Due to the n-privacy property of the decomposition, there is a simulator S_e that on input ϕ and y produces an output $(\{w_i, R_i\}_{i \in \{e, e+1, \ldots, e+(n-1)\}}, , y_{e-1})$ distributed as in the protocol. The simulator S now sets w_{e-1} and R_{e-1} as strings of corresponding lengths that contain only zeroes. Now, S can produce commitments $c_i = \mathsf{Comm}(R_i, w_i)$ for all $i = 0, \ldots, n$ and send $a = (y_i, c_i)_{0 \le i \le n}$. Clearly, the triple (a, e, z_e) has the same distribution as in a real conversation, as z_e can also be easily computed. Hence, the $(n+1)$-ZKBoo-protocol has the the special honest-verifier ZK property. □

In the last part, we analyze the *soundness error* of the $(n+1)$-ZKBoo protocol which can be directly derived from special soundness. Briefly speaking, the soundness error can be summarized as the probability of a cheating prover to trick a honest verifier to accept the protocol on a value $y \notin L_\phi$

More formally, the soundness error δ is defined as;

$$\max_{y \notin L_\phi, \mathcal{P}'} \{[\Pr((\mathcal{P}', \mathcal{V})(y) = \mathtt{accept}]\},$$

where $\mathcal{P}'$ is some cheating prover. As the challenge e is chosen uniformly at random from a set of cardinality $n+1$, the $n+1$-special soundness implies a soundness error of $\delta \le (n+1-1)/(n+1) = 1 - \frac{1}{n+1}$, as for $y \notin L$, there are at most $n+1-1$ accepting conversations for each a.

Let $\phi \colon \mathbb{K}^k \to \mathbb{K}^\ell$ be a function that can be expressed by an $(\lceil n/2 \rceil + 1, n)$ order decomposition with N strongly non-interfering gadgets such that $N_\otimes$ of them are balanced multiplication gadgets defined in Section 4. In order to attain soundness error $2^{-\sigma}$ we need to repeat the t-ZKBoo protocol k_n times such that,

$$\begin{aligned} 2^{-\sigma} &\ge (1 - \frac{1}{n+1})^{k_n} \Leftrightarrow \\ k_n &\ge -\sigma \cdot [\log(1 - \frac{1}{n+1})]^{-1}. \end{aligned} \tag{2}$$

Similar to [GMO16], the number of bits needed for the opened views is

$$-\sigma \cdot [\log(1-\frac{1}{n+1})]^{-1} \cdot \left(\left\lceil\frac{n}{2}\right\rceil+1\right) \cdot [\log(|\mathbb{K}|)(k+\ell+N_a)+\kappa]\,,$$

where κ is the security parameter to generate random tapes.

Theorem 3. *The* $(n+1)$*-ZKBoo protocol satisfies the* $(n-(\lceil n/2\rceil+1), \lceil n/2\rceil+1)$*-SNI notion given in Definition 3.*

Proof. As shown in Proposition 2, the evaluation protocol of the $(\lceil n/2\rceil+1, n+1)$-decomposition is n-SNI. As we open exactly $\lceil n/2\rceil+1$ computed shares, the $(n+1)$-ZKBoo protocol is still $n-(\lceil n/2\rceil+1)$-secure. $\square$

5.1 Experimental Results

In this section, we analyze the $(n+1)$-ZKBoo protocol as introduced in the previous section and compare it to other instatiations from the literature that have not been adjusted to achieve SNI security. The simulation of the traces is generated by evaluating Π_ϕ using a $(3,5)$-decomposition for a set of random inputs x and collecting the output shares of each node. Observe that the collected traces correspond to $w^{(j)}$ for all gadgets $1 \leq j \leq d$. We denote the ℓ^{th} trace (corresponding to the ℓ^{th} evaluation of the protocol) by $t_\ell = \{w_j^{(i)} \mid \text{ for all } i \in [0,n] \text{ and } j \in [0,d]\}$. Moreover, we assume that $e=0$ and collect the views of the $w_0, \ldots, w_{\lceil n/2\rceil}$ to reflect the opened views as *free probes*.

Using the synthetically generated traces, we perform a simple leakage detection test by the test vector leakage assessment (TVLA) as proposed by Gilbert et al. [GGJR$^+$11]. TVLA allows to detect leakages at specific orders and comes in two flavors: the specific variant and the non-specific variant. For the non-specific variant, two different sets of side-channel traces are collected by processing either a fixed input or a random input under the same conditions in a random pattern. The specific variant collects random traces and sorts them according to a distinguishing function. The advantage of the specific version is that the distinguishing function pinpoints specific leakages and can easily be turned into a side-channel attack. After collecting the traces, the means (μ_0, μ_1) and standard deviations (σ_0, σ_1) for two sets are calculated in both variants. Welch's t-test is computed as in Equation (3) where n_0 and n_1 denote the number of traces for the two distinguished sets, respectively.

$$t = \frac{\mu_f - \mu_r}{\sqrt{(\sigma_f^2/n_f)+(\sigma_r^2/n_r)}}. \tag{3}$$

Our analysis applies the specific t-test, where the classification of the traces relies on the opened views. Thus, we manage to experimentally verify that the opened values can be used to classify traces in a meaningful way, which can then be used to recover the secret. We start with $(n+1)$-ZKBoo with $n+1=3$, where the scheme is equivalent to the original ZKBoo protocol given in [GMO16]. As given in Section 3, ZKBoo violates the privacy notion due to the opened views. Using our experimental setup, we target a multiplication gadget whose index is denoted by α and perform the classification as follows:

$$\mathcal{T}_0 = \{t_i | w_0^{(\alpha)} \oplus w_1^{(\alpha)} = 0\} \text{ and } \mathcal{T}_1 = \{t_i | w_0^{(\alpha)} \oplus w_1^{(\alpha)} = 1\}$$

Remark that the corresponding views $w_0^{(\alpha)}$, $w_1^{(\alpha)}$ represent the opened shares during the challenge phase. Using the two sets of traces $\mathcal{T}_0$ and $\mathcal{T}_1$, we perform the t-test as given in Equation (3). The result of ZKBoo and the clear leakage can be seen in Figure 3.

Next we analyze the 5-ZKBoo protocol which uses a $(3, 5)$-circuit decomposition. As given in Section 4, the scheme is proven to be secure against first order attacks with three opened shares. We adapt the t-test and perform the classification as follows:

$$\mathcal{T}_0 = \{t_i | w_0^{(\alpha)} \oplus w_1^{(\alpha)} \oplus w_2^{(\alpha)} = 0\} \text{ and } \mathcal{T}_1 = \{t_i | w_0^{(\alpha)} \oplus w_1^{(\alpha)} \oplus w_2^{(\alpha)} = 1\},$$

where $w_0^{(\alpha)}$, $w_1^{(\alpha)}$ and $w_2^{(\alpha)}$ represent the opened shares during the challenge phase that correspond to the targeted gadget. The clear leakage in Figure 3 diminishes as seen in Figure 7a.

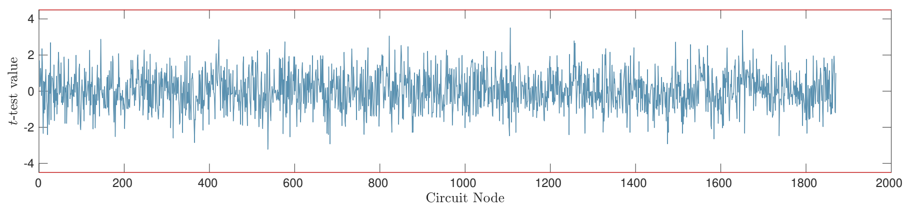
To compare our approach with previous unprotected approaches, we apply the same test while opening all views but one, as introduced in [KKW18], where a challenge e requires opening the views $(w_i)_{i \in [0,n],\ i \neq e}$. Using classification $\mathcal{T}_a = \{t_i | w_0^\alpha \oplus w_1^\alpha \oplus w_2^\alpha \oplus w_3^\alpha = a\}$ we can see the resulting clear leakages in Figure 7b.

6 Zero-Knowledge for Post Quantum Signature Schemes

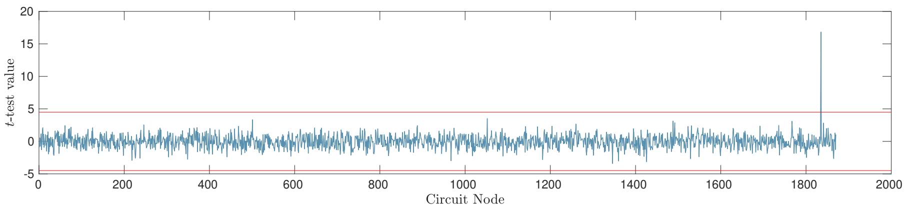
In this section, we describe the application of the $(n + 1)$-ZKBoo-protocol as given in Figure 6 with its performance analysis. First off all, we remark that the proposed circuit decomposition brings no overhead to the previous approaches, since we are using the same gadgets with a different technique. Therefore, the number of gadgets in each branch are the same as for the circuit decomposition defined in [IKOS09] or for the MPC-in-the-head idea defined in [KKW18].

6.1 Picnic Scheme using $(n + 1)$-ZKBoo

In this section we provide a variation of the Picnic signature scheme that can build upon our $(n + 1)$-ZKBoo protocol. This scheme was introduced by Chase et al. [CDG+17]. The fundamental component of the scheme is a zero knowledge proof using ZKBoo. The first version of the scheme uses an improved and optimized version of ZKBoo called ZKB++. The scheme uses the LowMc [ARS+15] block cipher as its underlying symmetric primitive (or ϕ in our notation). LowMc is a flexible block cipher with low AND depth especially suited for Secure Multi-Party Computation, Zero-Knowledge Proofs, or Fully Homomorphic Encryption.



(a) TVLA-based classification using $\mathcal{T}_i$ where $i = w_0^\alpha \oplus w_1^\alpha \oplus w_2^\alpha$.



(b) TVLA-based classification using $\mathcal{T}_i$ where $i = w_0^\alpha \oplus w_1^\alpha \oplus w_2^\alpha \oplus w_3^\alpha$.

Fig. 7: First order leakage analysis of a multiplication gadget in $(n+1)$-ZKBoo reveals no leakage with three opened shares (a), but is vulnerable with four opened shares (b).

To make the interacitve protocols ZKBoo/ZKB++ non-interactive, two methods are used: The Fiat-Shamir transformation [FS86] or the Unruh transformation [Unr15]. We do not focus on the transformations in this work, since our idea is to modify *only* the underlying circuit decomposition. Another circuit decomposition using higher orders of decomposition has been proposed [KKW18]. As the higher order decompositions result in increased soundness error, the authors proposed a preprocessing phase to compensate for this soundness error. We remark that our soundness error and number of required repetitions in Table 1 can be further optimized using that preprocessing phase. As a result, we can replace the underlying MPC-in-the-head protocol with the one defined in Section 4 and create a set of parameters as in Table 1 where L1, L3 and L5 represents the 128,192 and 256-bit level security for the scheme.

As seen from the Table 1 we can replace the underlying MPC-in-the-head paradigm with a small cost to gain probing security. The first analogy can be done by observing the required number of repetitions in order to achieve the desired soundness error.

To achieve probing security for L1, L2 and L3 we need to change the underlying circuit decomposition. The required number k_n can be calculated using Equation (2) and the summary can be found in Table 1. Due to the nature of the soundness, the required number of repetitions to obtain the appropriate soundness error increases with the decomposition order. Thus the higher number of shares implies improved soundness error at the cost of increased signature size. For example, to achieve first order protection ZKBoo using $(2,3)$ circuit

Table 1: The parameter set for the proposed circuit decomposition and the comparison between the scheme that uses ZKBoo (2,3) circuit decomposition with the probing security order $t_{\mathcal{A}}$ and with the required number of repetitions k_n.The Full version of the table can be found in Appendix C.

Parameter Set	Decomp.	$t_{\mathcal{A}}$	σ	k_n
picnic-L1-FS/UR	(2,3)	0	128	219
picnic-L1-FS/UR	(3,5)	1	128	398
picnic-L1-FS/UR	(4,7)	2	128	576
picnic-L1-FS/UR	(5,9)	3	128	745
picnic-L3-FS/UR	(2,3)	0	192	329
picnic-L3-FS/UR	(3,5)	1	192	597
picnic-L3-FS/UR	(4,7)	2	192	864
picnic-L3-FS/UR	(5,9)	3	192	1130
picnic-L5-FS/UR	(2,3)	0	256	438
picnic-L5-FS/UR	(3,5)	1	256	796
picnic-L5-FS/UR	(4,7)	2	256	1152
picnic-L5-FS/UR	(5,9)	3	256	1507

decomposition should be replaced with 5-ZKBoo using $(3, 5)$ circuit decomposition which results in 80% more repetitions. Similarly the cost of second order protection is 40% more repetition with respect to first order protection.

Secondly we can analyze the signature sizes and corresponding overhead. In this analysis we focus on the signature size of ZKBoo in order to make a fair comparison since ZKB++ improvements can be applied independent of the circuit decomposition structure. First we remark the signature size of ZKBoo. Assume that c the size of the commitments c_i and s denotes the size of the randomness in bits used for each commitment (Commit phase in Figure 6). The ZKBoo signature for a function $\phi\colon \mathbb{K}^k \to \mathbb{K}^\ell$ is given by [CDG+17]:

$$\begin{aligned}
|p_z| &= k_z[3(|y_i| + |c_i|) + 2((\log(|\mathbb{K}|)(k + \ell + N_\otimes)) + |R_i| + s)\\
&= k_z[3\log(|\mathbb{K}|) + c) + 2(\log(|\mathbb{K}|)(k + \ell + N_\otimes) + \sigma + s)]\\
&= k_z[3(\log(|\mathbb{K}|)(3c + 2\sigma + 2s + \log(|\mathbb{K}|)(5\ell + 2n + 2N_\otimes))
\end{aligned}$$

where $k_z = \lceil \sigma(\log(3) - 1)^{-1} \rceil$ is the number of required reputations to achieve the desired security order σ. Using the same idea we can calculate the signature size of $(n+1)$-ZKBoo as below. To the sake of readability we denote $n+1$ by u and $\lceil n/2 \rceil + 1$ by v:

$$\begin{aligned}
|p| &= k_n[u(|y_i| + |c_i|) + v((\log(|\mathbb{K}|)(k + \ell + N_\otimes)) + |R_i| + s)\\
&= k_n[u\log(|\mathbb{K}|) + c) + v(\log(|\mathbb{K}|)(k + \ell + N_\otimes) + \sigma + s)]\\
&= k_n[u(\log(|\mathbb{K}|)(uc + v\sigma + us + \log(|\mathbb{K}|)((u + v)\ell + vn + vN_\otimes))
\end{aligned}$$

where k is the required repetitions as defined in Section 5. Clearly the main the natural overhead of the higher decomposition given as the number of opened

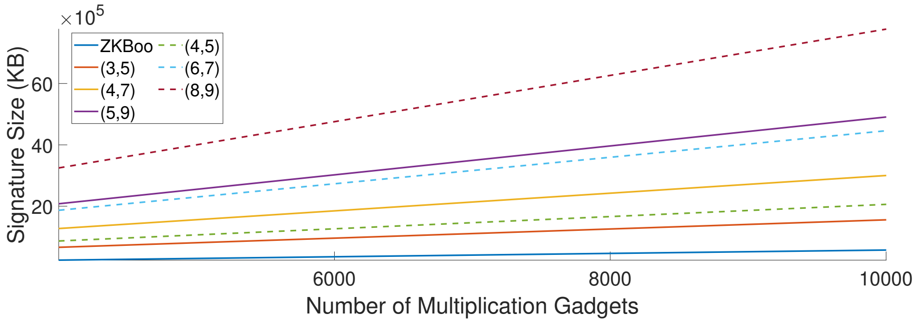


Fig. 8: The Signature size comparison with respect to number of multiplication gadgets. Dashed lines represents the circuit decomposition defined by [KKW18].

views such that ZKBoo needs 2 while out decomposition requires $\lceil n/2 \rceil + 1$ views. Thus the cost of replacing the underling (2,3)-ZKBoo decomposition with (3,5)-circuit decomposition is around doubling the size of the signature.

In order to visualize this, we use the $|p|$ formula to compare various protocols. We assume that 128-bit security is required ($\sigma = 128$) and ϕ function is selected as LowMc using 128-bit key such that $\phi : \mathbb{F}^{128} \to \mathbb{F}^{128}$. As given in Figure 8 the size of the signature naturally increases with the circuit decomposition order. However, the size is still smaller than the signature size of [KKW18] as a result of opening only $\lceil n/2 \rceil + 1$ views instead of n views.

In order to compare the running times of ZKBoo with $(n+1)$-ZKBoo, we give an approximate formula involving the running time T_{rand} to generate a random tape, the running time T_{comm} to compute a single commitment, and the running time T_{mult} of a single multiplication gate. In the original ZKBoo, the prover first creates 3 random tapes and later computes the commitment to 3 shares. Within the evaluation of $\Pi_\phi(x)$, each multiplication gate of the original circuit for ϕ is replaced by 3 multiplication gates in the $(2,3)$-decomposion. As there are 3 branches, the total running time of the prover is proportional to $3 \cdot (T_{\mathsf{rand}} + T_{\mathsf{comm}} + 3N_{\otimes}T_{\mathsf{mult}})$, where $N_{\otimes}$ is the number of multiplication gates in the circuit computing ϕ. The verifier in ZKBoo just needs to verify the computation of 2 branches. Hence, its running time is proportional to $6N_{\otimes}T_{\mathsf{mult}}$.

In $(n+1)$-ZKBoo, the prover creates $n+1$ random tapes and computes $n+1$ commitments. Furthermore, within each of the $n+1$ branches, each multiplication gate of the original circuit for ϕ is replaced by $n+2$ multiplication gates on average in the $(\lceil n/2 \rceil + 1, n+1)$-decomposition (see Section 4), as $2 \cdot [(n+1)(n+2)/2]$ multiplications are computed in total. Hence, the total running time of the prover is proportional to $(n+1) \cdot (T_{\mathsf{rand}} + T_{\mathsf{comm}} + (n+2)N_{\otimes}T_{\mathsf{mult}})$, where $N_{\otimes}$ is the number of multiplication gates in the circuit computing ϕ. The verifier in $(n+1)$-ZKBoo just needs to verify the computation of $\lceil n/2 \rceil + 1$ branches. Hence, its running time is proportional to $(\lceil n/2 \rceil + 1) \cdot (n+2)N_{\otimes}T_{\mathsf{mult}}$.

Assuming that the running time of $N_{\otimes}T_{\mathsf{mult}}$ dominates T_{rand} and T_{comm}, we have a multiplicative overhead of $(n+1)(n+2)/9$ for the prover and $(\lceil n/2 \rceil +$

$1)(n+2)/6$ for the verifier. In the case of $n+1=5$, this is an overhead of $30/9 \approx 3.34$ for the prover and 3 for the verifier.

7 Conclusion

As MPC-in-the-head based protocols are entering real-world applications, their susceptibility to side-channel attacks becomes relevant. In this work we have shown that current MPC-in-the-head protocols are indeed susceptible to SCA. Side channel adversaries can simply use the information of the revealed shares and combine it with information leaked via some side channel to recover secret intermediate states. Hence, currently proposed MPC-in-the-head protocols are just as susceptible to SCA as other cryptosystems.

However, as popular side-channel countermeasures also build on MPC principles, we show how to adapt MPC-in-the-head protocols so that they become side-channel resistant. By analyzing MPC-in-the-head protocols using the strong non-interference property, we first show that current proposals do not achieve strong non-interference. We also show how this lack of protection can be abused by side channel adversaries. We then show, using the example of ZKBoo, that MPC-in-the-head protocols can be adapted to fulfil SNI by adjusting parameter choices. Thus, changes at the protocol level suffice to achieve strong non-interference and allow the protection of MPC-in-the-head protocols right at the protocol level, without deeper changes to the implementations.

We have shown that MPC-in-the-head protocols lend themselves to be easily protected against side-channel attacks. By simply adjusting the parameters of the underlying MPC protocol, side channel resistance can be achieved. We practically verify this resistance and show that the resulting overheads are comparably low.

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *FOCS*, pages 474–483. IEEE Computer Society, 2014.

[BBC+19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and Francois-Xavier Standaert. maskverif: Automated verification of higher-order masking in presence of physical defaults. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *Computer Security – ESORICS 2019*, pages 300–318, Cham, 2019. Springer International Publishing.

[BBD$^+$15a] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. Cryptology ePrint Archive, Report 2015/506, 2015.

[BBD$^+$15b] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. *IACR Cryptology ePrint Archive*, 2015:506, 2015.

[BBD$^+$16] Gilles Barthe, Sonia Belaïd, Francois Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 116–129, New York, NY, USA, 2016. ACM.

[BGI$^+$18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 321–353. Springer, 2018.

[CCJ$^+$16] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[CDG$^+$17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM Conference on Computer and Communications Security*, pages 1825–1842. ACM, 2017.

[Cha19] André Chailloux. Quantum security of the fiat-shamir transform of commit and open protocols. *IACR Cryptology ePrint Archive*, 2019:699, 2019.

[CJRR99a] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CJRR99b] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer, 1999.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. *Unifying Leakage Models: From Probing Attacks to Noisy Leakage.*, pages 423–440. Springer, 2014.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.

[Dwo15] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.

[FGMDP^{+}18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, Aug. 2018.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[GGJR^{+}11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, pages 51–60. IEEE Computer Society, 2012.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1069–1083, Austin, TX, August 2016. USENIX Association.

[Gol07] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

[GSE20] Tim Gellersen, Okan Seker, and Thomas Eisenbarth. Differential power analysis of the picnic signature scheme. Cryptology ePrint Archive, Report 2020/267, 2020. `https://eprint.iacr.org/2020/267`.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 525–537, New York, NY, USA, 2018. Association for Computing Machinery.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KZ20] Daniel Kales and Greg Zaverucha. Improving the performance of the picnic signature scheme. Cryptology ePrint Archive, Report 2020/427, 2020. `https://eprint.iacr.org/2020/427`.

[NIS20a] NIST. *Post-Quantum Cryptography – Round 1 Submissions*, 2017 (accessed April 25, 2020).

[NIS20b] NIST. *Post-Quantum Cryptography – Round 2 Submissions*, 2019 (accessed April 25, 2020).

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings*, pages 413–427. Springer, 2010.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[TOS10] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on aes, and countermeasures. *Journal of Cryptology*, 23(1):37–71, 2010.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 755–784. Springer, 2015.

[YF14] Yuval Yarom and Katrina Falkner. Flush+reload: A high resolution, low noise, l3 cache side-channel attack. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 719–732, San Diego, CA, August 2014. USENIX Association.

A Details on ZKBoo

Definition 5 ((2,3) circuit decomposition). *A (2,3)-decomposition for the function ϕ is the set of functions;*

$$\mathcal{D} = \{\mathsf{Share}, \mathsf{Output}_1, \mathsf{Output}_2, \mathsf{Output}_3, \mathsf{Rec}\} \cup \mathcal{F}$$

where Share, Rec and Output_i as defined in Section 4. Let Π^*_ϕ be the algorithm described in Figure 5 with $n = 2$, we have the following definitions.

– ***Correctness:*** *We say that $\mathcal{D}$ is correct if $Pr[\phi(x) = \Pi^*_{\phi(x)}] = 1$ for all $x \in X$. The probability is over the choice of the random tapes R_i.*
– ***Privacy**: We say that $\mathcal{D}$ has 2-privacy if it is correct and for all $e \in [0,2]$ there exists a PPT simulator S_e such that $((k_i, w_i)_{i\in\{e,e+1\}}, y_{e+2})$ and $S_e(\phi, y)$ have the same probability distribution for all $x \in X$.*

B Algorithms and Proofs

Proof of Theorem 2

Proof. In order to prove the theorem, we show that every set of intermediate variables with $t \leq n$ elements can be simulated by a set of input shares $(x)_{i\in I}$

Algorithm 2 *Balanced* Multiplication Gadget

```
Input: The shares $(x_0, \ldots, x_n)$ and $(y_0, \ldots, y_n)$.
Output: The vector of shares of $xy$ as $(z_0, \ldots, z_n)$.
 1: for $0 \leq i \leq n$ do
 2:     for $i < j \leq n$ do
 3:         $r_{i,j} \leftarrow rand()$                          ▷ $r_{i,j} \in_R \mathbb{K}$
 4:         $r_{j,i} = (x_i y_j \oplus r_{i,j}) \oplus x_j y_i$
 5: for $0 \leq i \leq n$ do
 6:     $z_i \leftarrow x_i y_i$
 7:     for $0 \leq j \leq n$ and $j \neq i$ do
 8:         if $j \in J_i$ then
 9:             $z_i \leftarrow z_i \oplus r_{j,i}$                ▷ Denoted by $z_{i,j}$
10:         else
11:             $z_i \leftarrow z_i \oplus r_{i,j}$                ▷ Denoted by $z_{i,j}$
12: return $(z_0, \ldots, z_n)$
```

Algorithm 3 RefreshMask Gadget[BBD+16]

```
Input: The vector of shares $(x_0, \ldots, x_n)$.
Output: The vector of shares of $x$ as $(x'_0, \ldots, x'_n)$.
 1: for $0 \leq i \leq n$ do
 2:     $x'_i \leftarrow x_i$
 3: for $0 \leq i \leq n$ do
 4:     for $i < j \leq n$ do
 5:         $r \leftarrow rand()$                                ▷ $r \in_R \mathbb{K}$
 6:         $x'_i \leftarrow x'_i \oplus r$
 7:         $x'_j \leftarrow x'_j \oplus r$
 8: return $(x'_0, \ldots, x'_n)$
```

such that $|I| \leq t$. Let us first classify the variables. The intermediate variables are x_i, $r_{i,i+j}$, $a_{i,i+j}$ and $b_{i+j,i}$ and the outputs are x'_i (or $x'_{i,i+\lceil n/2 \rceil}$).

After this now we can define I as follows: for each probed variable x_i, $r_{i,i+j}$ and $a_{i,i+j}$ add i to I and $b_{i+j,i}$ add $i+j$ to I. It is clear that I contains at most t elements since each probed value adds at most one index to I.

Now we can define the simulator. For all $i \in I$ the simulator can sample all $r_{i,i+j}$ for for $j \in [0, \lceil n/2 \rceil]$ and compute all partials sums $a_{i,i+j}$ and $b_{i+j,i}$ and thus the output x'_i.

Now we need to focus on the simulation of the output shares x'_i such that $i \notin I$. Observe that $i \notin I$ means that the any random value in the partial sum of x'_i is not probed and does not involved in a partial sum of it. Hence we can simulate x'_i by a uniformly random value. As a result any set of t probed intermediate variables and any subset $\mathcal{O} \subset [0, n]$ can simulated by $x_{|I}$ such that $|I| \leq t$. □

C Parameters set for Picnic

Table 2: The parameter set for the proposed circuit decomposition and the comparison between the scheme that uses standard (2,3) circuit decomposition with the probing security order $t_{\mathcal{A}}$ and with the required number of repetitions k. The LowMc parameters are key size L_n, number of s-boxes L_s, and number of rounds L_r. The hash functions in L1 specification are all based on the SHAKE-128 while L3 and L5 are based on SHAKE-256 SHA-3 functions [Dwo15] and ℓ_h represents the output length of hash functions.

Parameter Set	Decomp.	$t_{\mathcal{A}}$	σ	L_n	L_s	L_r	k	Hash/KDF	ℓ_h
picnic-L1-FS/UR	(2,3)	0	128	128	10	20	219	SHAKE128	256
picnic-L1-FS/UR	(3,5)	1	128	128	10	20	398	SHAKE128	256
picnic-L1-FS/UR	(4,7)	2	128	128	10	20	576	SHAKE128	256
picnic-L1-FS/UR	(5,9)	3	128	128	10	20	745	SHAKE128	256
picnic-L3-FS/UR	(2,3)	0	192	192	10	30	329	SHAKE256	384
picnic-L3-FS/UR	(3,5)	1	192	192	10	30	597	SHAKE256	384
picnic-L3-FS/UR	(4,7)	2	192	192	10	30	864	SHAKE256	384
picnic-L3-FS/UR	(5,9)	3	192	192	10	30	1130	SHAKE256	384
picnic-L5-FS/UR	(2,3)	0	256	256	10	38	438	SHAKE256	512
picnic-L5-FS/UR	(3,5)	1	256	256	10	38	796	SHAKE256	512
picnic-L5-FS/UR	(4,7)	2	256	256	10	38	1152	SHAKE256	512
picnic-L5-FS/UR	(5,9)	3	256	256	10	38	1507	SHAKE256	512