Formal security analysis of MPC-in-the-head zero-knowledge protocols

Nikolaj Sidorenco
Dept. of Computer Science
Aarhus University
Aarhus, Denmark
sidorenco@cs.au.dk

Sabine Oechsner
Dept. of Computer Science
Aarhus University
Aarhus, Denmark
oechsner@cs.au.dk

Bas Spitters
Concordium Blockchain Research Center
Aarhus University
Aarhus, Denmark
spitters@cs.au.dk

2021 IEEE 34th Computer Security Foundations Symposium (CSF) | 978-1-7281-7607-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSF51468.2021.00050

***Abstract*—Zero-knowledge proofs allow a *prover* to convince a *verifier* of the veracity of a statement without revealing any other information. An interesting class of zero-knowledge protocols are those following the MPC-in-the-head paradigm (Ishai et al., *STOC '07*) which use secure multiparty computation (MPC) protocols as the basis. Efficient instances of this paradigm have emerged as an active research topic in the last years, starting with ZKBoo (Giacomelli et al., *USENIX '16*). Zero-knowledge protocols are a vital building block in the design of privacy-preserving technologies as well as cryptographic primitives like digital signature schemes that provide post-quantum security.**

This work investigates the security of zero-knowledge protocols following the MPC-in-the-head paradigm. We provide the first machine-checked security proof of such a protocol on the example of ZKBoo. Our proofs are checked in the EasyCrypt proof assistant. To enable a modular security proof, we develop a new security notion for the MPC protocols used in MPC-in-the-head zero-knowledge protocols. This allows us to recast existing security proofs in a black-box fashion which we believe to be of independent interest.

I. Introduction

Zero-knowledge proofs [1] allow a party, the prover, to convince another party, acting as the verifier, of the veracity of some statement without revealing anything else. This seemingly paradoxical primitive lies at the heart of many modern privacy-preserving technologies, and more generally is a crucial cryptographic building block for applications like digital signature schemes.

One approach to constructing zero-knowledge proofs has gained particular attention over the last years: the MPC-in-the-head paradigm of Ishai et al. [2] which uses secure multiparty computation (MPC) protocols in a surprising way as the building block. Consider the setting where a prover holds the pre-image x of a public one-way function f and has published $y = f(x)$. To convince the verifier that they indeed know x corresponding to y, the prover will first split the secret x into random shares $x_1, \ldots, x_n$ such that $\sum_i x_i = x$. The prover then emulates an MPC protocol "in their head", with the catch that the protocol performs a distributed computation of $f(x)$ with shares $x_1, \ldots x_n$ as inputs. This emulation yields one transcript of the protocol execution per party. Prover and verifier can then interact to reveal a subset of transcripts, which the prover can check for consistency. If the consistency check succeeds, then the verifier will be convinced that the prover knows x. Intuitively, this does not leak any information about x if the MPC protocol is secure against insider corruption of some parties and not too many transcripts are revealed.

While at first believed to be of purely theoretical interest, the MPC-in-the-head paradigm was subsequently shown to be of practical relevance [3]. Combined with the Fiat-Shamir heuristic [4], one can moreover obtain efficient digital signature schemes from such zero-knowledge proofs. In fact, Picnic, a successful contender for the NIST post-quantum cryptography standardization competition [5], follows this design pattern. Moreover, multiple efficiency improvements have been proposed recently [6], [7]. Given the standardization potential of this approach, it is natural to ask to formally verify such constructions.

A. Our Contributions

In this work, we investigate the security of MPC-in-the-head type zero-knowledge proofs like ZKBoo [3], Picnic [5], [8], KKW [9], and Banquet [7].

- We provide the first machine-checked security proof of a zero-knowledge protocol following the MPC-in-the-head paradigm. Our mechanization studies the ZKBoo protocol [3] and is done in the EasyCrypt proof assistant [10]. We show that our formalisation is security assuming the existence of a perfectly hiding commitment scheme. This slightly deviates from the implementation of ZKBoo, which is instantiated with a computational hiding commitment scheme. Interestingly, protocols following the MPC-in-the-head paradigm use MPC protocols as a building block in a bigger construction rather than as a goal, and we are not aware of any other machine-checked proof with this property.
- To enable a modular security proof, we develop a new security notion for the MPC protocols in question which is of independent interest. The new notion enables us to give black-box security proofs of MPC-in-the-head zero-knowledge protocols.

Our starting point is the ZKBoo protocol by Giacomelli et al. [3] as a representative of this protocol class. From a technical perspective, this class of protocols is an interesting challenge due to the unconventional combination of complex primitives like MPC and zero-knowledge proofs. Based on the observation that modularity of existing constructions currently does not carry over to modularity of proofs, we propose to use a refined notion of the MPC protocol (called *decomposition* protocol, to keep with the ZKBoo terminology). This new decomposition notion then allows us to define black-box transformations from decomposition to Σ-protocols, a special class of zero-knowledge protocol. To demonstrate the generality of this approach, we recast existing protocols in this style. On a conceptual level, this clear separation between decomposition and transformation to Σ-protocol improves the understanding of the different optimization strategies, and can hopefully help find new ones. With a clear proof strategy set up, we then proceed to mechanize the security proof in EasyCrypt. The EasyCrypt code is available at https://github.com/Nsidorenco/Decomposition-zk

B. Outline

Section II presents the necessary background for the rest of this work. The MPC-in-the-head paradigm is presented in Section III, and we discuss moreover the ZKBoo protocol and its existing security proof as an example. In Sections IV and V we present our new decomposition notion and demonstrate the black-box construction of a Σ-protocol from it. Further protocols, and their relation to our formalization, are discussed in Section VI. Section VII presents our EasyCrypt formalization of the ZKBoo protocol. Related work is discussed in Section VIII before we discuss future work and conclude in Section IX and X.

II. Preliminaries

This section presents some cryptographic concepts and notations.

Notation We let $[n]$ denote the set $\{1, 2, \ldots, n\}$, for any given integer n and let $|A|$ denote the cardinality of the set A.

Given two probability distributions X and Y we define the statistical distance between them as

$$\mathbf{SD}(X, Y) = \frac{1}{2}\sum_i |\Pr[X = i] - \Pr[Y = i]|.$$

Two families of random variables $X = \{X_k\}$, $Y = \{Y_k\}$ indexed by bit-strings $k \in \{0,1\}^*$ are said to be *perfectly indistinguishable* if $X_k = Y_k$ for all k. We write $X \sim Y$ for perfectly indistinguishable families X and Y. They are said to be *statistically indistinguishable* if there exists a negligible function $\epsilon(\cdot)$ such that for every $k, \mathbf{SD}(X_k, Y_k) \leq \epsilon(|k|)$. They are said to be *computationally indistinguishable* if there exists an efficient distinguisher D with a corresponding negligible function, such that for all k,

$$|\Pr[D(X_k) = 1] - \Pr[D(Y_k) = 1]\ | \leq \epsilon(|k|).$$

A. Commitments

A *commitment scheme* is a cryptographic primitive that allows a committer holding message m to convince a verifier of the following. Firstly, that some m was fixed at some point in time without revealing the value of m. This is done by sending a *commitment*, i.e. a token derived from m to the verifier. Second, the committer can later open the commitment to reveal m and convince the verifier that the message was not modified in the meantime.

Definition 1 (Commitment scheme): A *commitment scheme* consists of a tuple $(\mathtt{setup}, \mathtt{com}, \mathtt{cverify})$ of probabilistic algorithms with the following properties:

- Correctness: Let $ck \leftarrow \mathtt{setup}(1^\kappa)$. For all m and $(c, r) \leftarrow \mathtt{com}(ck, m)$, we have $\mathtt{cverify}(m, c, r) = \top$.
- Perfect hiding: Let $ck \leftarrow \mathtt{setup}(1^\kappa)$. For all m, m', the distributions $\mathtt{com}(ck, m)$ and $\mathtt{com}(ck, m')$ are identical.
- Computational binding: Let $ck \leftarrow \mathtt{setup}(1^\kappa)$, and c a commitment. Then for any adversary and message m, the probability of finding m, m', r, r', where $m \neq m'$, such that

$$\mathtt{cverify}(m, c, r) = \mathtt{cverify}(m', c, r') = \top$$

  is negligible.

Note that we limit ourselves to the above definition of perfectly hiding and computationally binding commitments. There are other notions in the literature.

B. MPC

A secure multiparty computation (MPC) protocol allows a set of n mutually distrusting parties $P_1, \ldots, P_n$ to compute a public function f of their private inputs $x_1, \ldots, x_n$. The function f is typically assumed to be represented as an arithmetic circuit. Security of MPC protocols can be studied with respect to different corruption models. In this work, we focus on passive security (also called honest-but-curious), where all protocol participants are assumed to follow the protocol specification but might try to derive additional information from the messages they receive. An MPC protocol is deemed *passively secure* if it provides

- Correctness: Parties learn the correct output $f(x_1, \ldots, x_n)$, and
- Privacy: Parties do not learn anything about the inputs of honest parties beyond what $f(x_1, \ldots, x_n)$ reveals.

We will denote by *view* the transcript of a protocol execution from the point of view of a party P_i, consisting of the input x_i, all messages P_i receives, as well as its random choices.

C. Zero-knowledge protocols

Zero-knowledge protocols [1] are a cryptographic primitive that allows a prover P to convince a verifier V of the veracity of a public statement, without revealing anything beyond that fact.

1) Σ-protocols: An important subclass of zero-knowledge protocols consists of the Σ-protocols [11]. A Σ-protocol is a zero-knowledge proof of knowledge for a relation R, i.e. it allows a prover to prove knowledge of a witness x for a public statement h in relation R.

Definition 2 (Σ-protocol): Let R be a relation. A Σ-*protocol* for R is an interactive protocol between a prover P and a verifier V, where P and V hold a common input h and P has additional secret input x with $R(h, x)$, with the following properties:

- The protocol has a special 3-move form (a, e, z) as shown in Fig 1.
- Completeness: If prover P is honest, i.e. $R(h, x)$ and P follows the protocol, then an honest verifier V will always accept.
- s-special soundness: Given s transcripts $(a, e_1, z_1), \dots, (a, e_s, z_s)$, an x' with $R(h, x')$ can be extracted from the transcripts.
- Special honest-verifier zero-knowledge: Assuming that the verifier is honest, there exists a simulator S that simulates transcripts such that real and simulated transcripts are statistically indistinguishable.

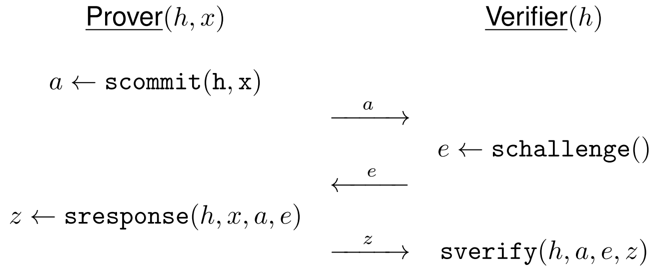
Prover(h, x)
Verifier(h)
$a \leftarrow \texttt{scommit(h, x)}$
a
$e \leftarrow \texttt{schallenge}()$
e
$z \leftarrow \texttt{sresponse}(h, x, a, e)$
z
$\texttt{sverify}(h, a, e, z)$

Fig. 1: Σ-Protocol overview

Interactive Σ-protocols can be made non-interactive and turned into digital signature schemes via the Fiat-Shamir transformation [4]. The idea is to replace the random challenge of the verifier by the output of a hash function on the statement to be proved and the first protocol message. This ensures that the prover chooses the first message before seeing the challenge. This transformation from proof of knowledge to signature was proven secure in the random oracle model by Pointcheval and Stern [12].

III. The MPC-in-the-head paradigm

Since the invention of the zero-knowledge concept, many approaches to constructing protocols were proposed. In recent years, the *MPC-in-the-head paradigm* (Ishai et al. [2]) has gained popularity. In this section, we briefly revisit the MPC-in-the-head paradigm as well as the ZKBoo protocol.

A. MPC-in-the-head-based zero-knowledge

To obtain a zero-knowledge protocol from an MPC protocol, the MPC-in-the-head paradigm proposes the following idea. Assume there is a public function ϕ and value y, and we want to prove knowledge of a witness x such that $\phi(x) = y$ in zero-knowledge. The value y could, for example, be the output of the SHA-256 hash function ϕ. As is standard in the MPC literature, we assume that ϕ is given in the form of a circuit.

- The prover P starts by secret sharing the private input x into inputs $x_1, \dots, x_n$ to virtual parties $P_1, \dots, P_n$. Assume that the circuit representation of ϕ is chosen such that it evaluates the function on such a shared input. The prover then runs an MPC protocol for evaluating ϕ on those shares "in their head". As a result, P obtains one protocol transcript for each party, also referred to as *views*. The prover then commits to all views and sends the commitments to the verifier V.
- The prover and verifier engage in an interactive protocol to select and open a random subset of committed views.
- The prover opens those commitments to reveal the requested views.
- The verifier checks for consistency of the opened views and accepts if they are consistent as well as valid openings of the commitments.

The crucial observation is that if the MPC protocol allows for local verifiability of views, then the above idea yields zero-knowledge protocols. While the MPC-in-the-head paradigm was initially believed to be of mostly theoretical interest, a series of recent works, starting with ZKBoo [3], showed it to be of practical relevance.

B. ZKBoo

We will now study the ZKBoo protocol as a concrete instance of the MPC-in-the-head paradigm. The ZKBoo protocol [3] was the first construction to show that the MPC-in-the-head paradigm [2] could be instantiated to yield a practically efficient protocol. The idea is to use a secret-sharing-based MPC protocol with three parties and a particular communication pattern as the basis: Each party P_i only sends messages to one of the other parties, namely their neighbour P_{i-1}. This pattern ensures that meaningful consistency checks can be performed given a pair of views of a protocol execution. The protocol operates on arithmetic circuits over a finite field $\mathbb{Z}_p$.

1) The Construction: For convenience, and to separate the MPC protocol from the Σ-protocol construction, the authors define *(2,3)-decomposition*. This is the view generation for an MPC protocol with three parties and privacy against passive corruption of two parties. This decomposition can then be combined with any commitment scheme to obtain a Σ-protocol for proving knowledge of a pre-image of a value y under a function ϕ.

a) (2,3)-Decomposition: Let ϕ be a function that is represented as a circuit with N gates. A (2,3)-decomposition for ϕ is defined as follows:

Definition 3 ([3]): A *(2,3)-decomposition* for a function ϕ is the set of functions $\mathcal{D} = \{\mathsf{Share}, \mathsf{Rec}, \phi_1^{(1)}, \dots, \phi_1^{(N+1)}, \dots, \phi_3^{(1)}, \dots, \phi_3^{(N+1)}, \mathsf{Output}_1, \mathsf{Output}_2, \mathsf{Output}_3\}$. Share is a surjective function splitting a single value into three values. The function ϕ_i^m computes the m'th gate of the circuit ϕ from the point of view of party i.

Output_i extracts the output value from party i. The output values of all parties can then be supplied to Rec, which constructs the shared output.

- (Correctness) is *correct* if $\Pr[\phi(\mathbf{x}) = \Pi_\phi^*] = 1$ for all $\mathbf{x} \in X$. The probability is computed over the choice of the random tapes $\mathbf{k}_i$.
- (Privacy) has *2-privacy* if it is correct and for all $e \in [3]$ there exists a PPT simulator S_e such that

Protocol Π^*_ϕ

Let $\phi\colon X \to Y$ be a function and $\mathcal{D}$ a related (2,3)-decomposition as defined in Def. 3.

Input: $\mathbf{x} \in X$

1) Sample random tapes $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$.
2) Compute $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \leftarrow \mathsf{Share}(\mathbf{x}; \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)$.
3) Let $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ be vectors with $N+1$ entries. Initialize $\mathbf{w}_i[0] = \mathbf{x}_i$, for all $i \in [3]$.
   - For $j = 1, \ldots, N$ and $i \in [3]$ compute: $\mathbf{w}_i[j] = \phi_i^{(j)}((\mathbf{w}_m[0..j-1], \mathbf{k}_m)_{m \in \{i, i+1\}})$.
4) Compute $\mathbf{y}_i = \mathsf{Output}(\mathbf{w}_i, \mathbf{k}_i)$ for $i \in [3]$.
5) Compute $\mathbf{y} = \mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3)$.

Output: $\mathbf{y} \in Y$.

Fig. 2: Protocol Π^*_ϕ describing how to use decomposition, used in Def. 3. Reproduced from [3].

$(\{\mathbf{k}_i, \mathbf{w}_i\}_{i \in \{e, e+1\}}, \mathbf{y}_{e+2})$ and $S_e(\phi, \mathbf{y})$ have the same probability distribution for all $\mathbf{x} \in X$.

The decomposition functions are implemented by ZKBoo as:

- $\mathsf{Share}(\mathbf{x}; \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)$ performs an additive secret sharing of $\mathbf{x}$ into three random shares $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ such that $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3$.
- $\mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3)$ outputs $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3$.
- The gate evaluation functions $\phi_i^{(j)}$ are defined as follows. Consider the j-th gate, and let a and b be its left and right input gates, resp. Then for $i \in [3]$, $\phi_i^{(j)}$ is defined as:
  - unary addition of α:
  $$\mathbf{w}_i[j] = \phi_i^{(j)}(\mathbf{w}_i[a]) = \begin{cases} \mathbf{w}_i[a] + \alpha & \text{if } i = 1 \\ \mathbf{w}_i[a] & \text{otherwise} \end{cases}$$
  - unary multiplication by α: $\mathbf{w}_i[j] = \phi_i^{(j)}(\mathbf{w}_i[a]) = \alpha \cdot \mathbf{w}_i[a]$
  - binary addition: $\mathbf{w}_i[j] = \phi_i^{(j)}(\mathbf{w}_i[a], \mathbf{w}_i[b]) = (\mathbf{w}_i[a] + \mathbf{w}_i[b])$
  - binary multiplication:
  $$\begin{aligned} \mathbf{w}_i[j] =& \phi_i^{(j)}(\mathbf{w}_i[a,b], \mathbf{w}_{i+1}[a,b]) \\ =& \mathbf{w}_i[a] \cdot \mathbf{w}_i[b] + \mathbf{w}_{i+1}[a] \cdot \mathbf{w}_i[b] \\ & + \mathbf{w}_i[a] + \mathbf{w}_{i+1}[b] + R_i(j) - R_{i+1}(j) \end{aligned}$$
  where $R_i(j)$ is sampled uniformly using $\mathbf{k}_i$.
- $\mathsf{Output}_i(\mathbf{w}_i, \mathbf{k}_i)$ selects the shares of the output wires of the circuit.

Specifically, the gate evaluation functions $\phi_i^{(j)}$ restrict communication between all parties of the protocol. Any of the ϕ_i functions can only take inputs from two distinct parties. By restricting all internal computations of the protocol to two parties we can freely reveal all inputs to one gate evaluation function, without revealing the reconstructed output. The secrecy of the reconstructed input is ensured by 2-privacy of decomposition.

b) ZKBoo protocol: Given the (2,3)-decomposition described above and a commitment scheme, the ZKBoo protocol proceeds to construct a Σ-protocol as shown in Fig. 3, following the MPC-in-the-head paradigm. The protocol is shown to be a Σ-protocol assuming the security of the commitment scheme and the (2,3)-decomposition.

2) Black-Box Security: We will now revisit the security proof of the ZKBoo construction. The proof of [3, Prop. 4.2] is not black-box as it relies on implementation specifics rather than on the security guarantees given by the decomposition and the commitment scheme.

a) Revisiting the ZKBoo security proof: To prove that the ZKBoo construction is a Σ-protocol, it is necessary to prove three properties: Completeness, 3-special soundness and special honest-verifier zero-knowledge. We stress that our findings do not affect the correctness of the existing security proof, but only its modularity and transferability to further constructions.

The *completeness* property is derived from the correctness of the commitment scheme in combination with correctness of the decomposition. There is, however, a subtle issue that prevents this proof step from being fully black-box: Correctness of the decomposition itself does not guarantee anything about the verifier's ability to verify the opened views. More specifically, correctness is a property of the protocol Π^*_ϕ derived from a decomposition $\mathcal{D}$ relating the *outputs* of $\Pi^*_\phi(x)$ and $\phi(x)$ for all x. This property lacks a statement about the *intermediate computation results*, i.e. the views of the decomposition resulting in the output, which is needed to reason about the verification procedure. Indeed, the standard correctness property in the MPC literature only guarantees correctness of the end result and not the intermediate computation steps[1]. Hence the security proof needs to revisit the concrete implementation of verification (recomputing the views in this case) and conclude that verification is indeed possible.

The *3-special soundness* property is a modified special soundness property that proves witness extraction given 3 transcripts (instead of the usual 2). The proof relies on multiple assumptions: First, the binding property of the commitment scheme is used to argue that the opened views are identical in the overlapping indices except with negligible probability. The next step invokes the reconstruction property of the specific secret sharing scheme used by ZKBoo to extract a potential input. This non-black-box step is necessary due to the lack of an explicit extractability guarantee of the decomposition notion. Correctness of the decomposition ensures that the extracted input is valid.

Finally, *special honest-verifier zero-knowledge* follows directly from 2-privacy of the decomposition and the hiding property of the commitment scheme, so this part is black-box.

b) Conclusion: As explained above, the ZKBoo security proof is not black-box, which seems to stem from an incomplete formalization of the required properties of the underlying MPC protocol. In the next sections, we will make a black-box construction and proof. To do so we modify the notion of

[1] Correctness of intermediate steps is of course shown during the proof, but this information is usually dropped in the final statement as it is not necessary for many applications.

ZKBoo protocol

The verifier and the prover have input $\mathbf{y} \in L_\phi$. The prover knows $\mathbf{x}$ such that $\mathbf{y} = \phi(\mathbf{x})$. A (2,3) decomposition of ϕ is given. Let Π^*_ϕ be the protocol related to this decomposition.

Commit:The prover does the following:

1) Sample random tapes $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$.
2) Run Π^*_ϕ and obtain the views $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ and the output shares $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$.
3) Commit to $\mathbf{c}_i = \mathtt{com}(\mathbf{k}_i, \mathbf{w}_i)$ for all $i \in [3]$.
4) Send $\mathbf{a} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$.

Prove: The verifier chooses an index $\mathbf{e} \in [3]$ and sends it to the prover. The prover answers to the verifier's challenge sending opening $\mathbf{c}_e, \mathbf{c}_{e+1}$ thus revealing $\mathbf{z} = (\mathbf{k}_e, \mathbf{w}_e, \mathbf{k}_{e+1}, \mathbf{k}_{e+1})$.

Verify:The verifier runs the following checks:

1) If the openings of commitments $\mathbf{c}_e, \mathbf{c}_{e+1}$ do not verify, output reject.
2) If $\mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \neq \mathbf{y}$, output reject.
3) If $\exists i \in \{e, e+1\}$ such that $\mathbf{y}_i \neq \mathsf{Output}_i(\mathbf{w}_i)$, output reject.
4) If $\exists j$ such that $\mathbf{w}_e[j] \neq \phi_e^{(j)}(\mathbf{w}_e, \mathbf{w}_{e+1}, \mathbf{k}_e, \mathbf{w}_{e+1})$, output reject.
5) Otherwise output accept.

Fig. 3: ZKBoo protocol, reproduced from [3].

decomposition. This formalization is not limited to ZKBoo, but captures a range of other protocols, as we will discuss in Section VI

IV. Decomposition protocols

Now that we understand why the decomposition notion of Giacomelli et al. [3] is not sufficient for a black-box security proof of the ZKBoo protocol, we will remedy this. This section proposes a new decomposition notion and explains how the (2,3)-decomposition of Giacomelli et al. relates to it. In Section V we will provide a black-box construction of the ZKBoo protocol from our decomposition notion.

A. Syntax and Security

Let us first consider the syntax. First of all, we combine the Share and $\phi_i^{(j)}$ functions into one `decompose` algorithm since they are no longer used separately. Next, remember that the black-box proof issues we discussed relate to the extractability of a witness from views as well as a lack of understanding of verification of the views. To mitigate these issues, we add a new `verify` algorithm to the decomposition notion. Finally, we observe that the optimizations of the ZKBoo protocol which we investigate in Section VI improve efficiency by not sending the full views in the last message of the Σ-protocol, but instead performing a reversible compression step. For this reason, we add the `compress` algorithm to our formalization. So, the syntax of a decomposition looks as follows:

Definition 4 (Decomposition protocols): Let n denote the number of parties. Let $\mathcal{C}$ and $\mathcal{R}$ be distributions and let $\leftarrow_R \mathcal{R}$ denote uniformly sampling an element from the distribution. A decomposition π is a collection of algorithms: $(\mathtt{decompose}, \mathtt{compress}, \mathtt{verify}, \mathtt{out}, \mathtt{rec})$ such that

- $\mathtt{decompose}(\phi, x, ks)$ takes a circuit ϕ with input x and a collection of random values and returns n views. We fix the distribution $\mathcal{R}$ as the universe of all random value inputs accepted by `decompose`.
- $\pi(vs, e)$ is a projection mapping a collection of n views to d views based on a challenge e.
- $\mathtt{compress}(v)$ is a compression function that transforms a view w into an alternative representation.
- $\mathtt{uncompress}(w)$ is the inverse of `compress`.
- $\mathtt{verify}(\phi, ws', e, ys)$ takes a circuit, d compressed views, a challenge, and n output shares and returns true/false,
- $\mathtt{out}(w[i])$ takes a view and returns the output share,
- $\mathtt{rec}(ys)$ takes a list of output shares and returns the output value of the circuit.

Our definition differs from the original by omitting the explicit communication pattern of the $\phi_i^{(j)}$ function. While the communication pattern is the mechanism enabling verification of views, we believe it is an implementation detail of `decompose` which is ensured by the security properties. By omitting the communication pattern we achieve a more succinct definition that allows for any arbitrary communication pattern provided the parties can still verify projected views without revealing information about the input.

After defining the syntax of a decomposition protocol, we will now express its security. We identify four properties of interest: verifiability, privacy, special soundness, and losslessness[2] of compression.

a) Verifiability: Verifiability captures that the views of a subset of parties in an honest execution of the protocol can be verified. This property subsumes and extends correctness of the underlying MPC protocol.

Definition 5 (Verifiability): For any fixed ϕ accepted by the decomposition, we say π is *verifiable* if for all challenges $e \in \mathcal{C}$ and inputs x,

$$\Pr[\text{verifiability\_game}(\phi, x, e)] = 1$$

[2] In EasyCrypt the term losslessness refers to sub-distributions with probability mass 1. In this work we refer to the terminology for compression functions, stating the compression function does not lose information.

where

```
$\text{verifiability\_game}(\phi, x, e) = \{$
    $rs \leftarrow_R \mathcal{R};$
    $ws \leftarrow \texttt{decompose}(c, x, ks);$
    $ys \leftarrow \texttt{map out } ws;$
    $y \leftarrow \texttt{rec}(ys);$
    $ws' \leftarrow \pi(ws\ e);$
    $valid \leftarrow \texttt{verify}(c, ws', ys);$
    $\textbf{return } valid \wedge \phi(x) = y$
$\}$
```

b) d-Privacy: The next property, d-privacy, captures the fact that a subset of views of size d does not reveal the input to the decomposition protocol. As is common in cryptography, this privacy property is stated using simulators. Note that the simulator is required to simulate not the parties' views obtained from the decompose function, but their compressed versions. Moreover, the simulator should be able to produce the output shares for all n parties which are indistinguishable from real output shares.

Definition 6 (d-Privacy): A decomposition π is said to be *d-private* if for all challenges $e \in \mathcal{C}$ and accepted circuits ϕ there exists a PPT simulator S_e such that

$$real(\phi, x, e) \sim \mathtt{S_e}(\phi, c(x))$$

where

```
$real(\phi, x, e) = \{$
    $rs \leftarrow_R \mathcal{R};$
    $ws \leftarrow \texttt{decompose}(c, x, ks);$
    $ys \leftarrow \texttt{map out } ws;$
    $\textbf{return } \pi(ws\ e, ys);$
$\}$
```

c) k-Special Soundness: Moreover, we require k-special soundness, meaning that given multiple partial (compressed) protocol views that are consistent with each other and verify, it is possible to extract a valid input to the protocol. In particular, given any subset of views of size k, we can extract a valid input to the protocol.

Definition 7 (k-Special Soundness):

A decomposition π has *k-special soundness* if there exists a PPT extractor `witness_extractor` such that for any k tuples of $(ws'_1, es_1, ys_1), \ldots, (ws'_k, es_k, ys_k)$

- If $es_1, \ldots, es_k$ are pairwise different, and
- if the projected views are pairwise consistent, i.e. $\forall i, j, v : v \in ws'_i \wedge v \in ws'_j \implies ws'_i[v] = ws'_j[v]$. Here $v \in ws'_i$ denotes that the view with index v (from the original set of views) is contained within the projection ws'_i
- if each set of projected views verifies, i.e. $\forall i, \texttt{verify}(\phi, ws'_i, es_i, ys_i) = true$,
- then $\Pr[\phi(\texttt{witness\_extractor}(c, \{ws'_i, es_i\}_{\forall i}) = \texttt{rec}(\texttt{map out}(ys_1))] = 1$.

d) Losslessness of compression: Finally, we require the compression function to be lossless and hence completely reversible.

Definition 8 (Lossless Compression): Let `compress` be a compression function with domain D. `Compress` is *lossless* if there exists an efficiently computable function `uncompress` such that for all $x \in D$, $\texttt{uncompress}(\texttt{compress}(x)) = x$.

e) Decomposition Security: Combining the properties above, we obtain the following security definition for decomposition protocols:

Definition 9 (Secure decomposition protocol): Let $k, d \in \mathbb{N}$. A decomposition protocol ϕ is *(k, d)-secure* if it has verifiability, d-Privacy, k-Special Soundness, and its decompression is lossless.

We note that the compression function and the losslessness property is not necessarily a property of a decomposition. For practical applications compression could be defined by itself. For this work compression is part of the decomposition, since it is, formally, easier to reason about compression when given direct access to the structure of the views.

B. Example: ZKBoo Decomposition Protocol

We now show that our new definition of a secure decomposition captures existing protocols by considering the example of ZKBoo. Further examples will be discussed in Section IV. The construction, recast in our syntax, looks as follows:

- $\mathcal{R}$ is the universe of all three element tuples (ks_1, ks_2, ks_3) where ks_i is a list of N random values.
- $\mathcal{C} = \{1, 2, 3\}$
- π outputs views with index e and $e + 1$.
- `out` and `rec` work exactly as before.
- `compress` and `uncompress`are unused and left as the identity function.
- `decompose` is a combination of Share and the gate computation functions $\phi_i^{(j)}$ from Section III-B. Concretely, the function corresponds to steps 2 and 3 in Fig. 2.
- `verify` performs the following checks:
  - The views are well-formed.
  - The output shares $ys[e]$, $y[e+1]$ are consistent with the output gate shares in the corresponding views $ws'[e+1]$, $ws'[e+1]$.
  - For $j = 1, \ldots, N$ and $i = 1, \ldots, 3$, $\phi_i^{(j)}(ws'[a], ws'[b], ws'[a], ws'[b]) = ws'[e][j]$. Here $ws'[e][j]$ denotes the share of gate j in view e.

Lemma 1: The construction described above is a secure decomposition for $d = 2$ and $k = 3$.

Proof: To show security, we need to prove verifiability, 3-special soundness, 2-privacy and losslessness of the compression.

Verifiability is an extension of the original correctness proof. Correctness concludes that the output shares reconstruct to the value of circuit evaluation, for any input. It is proven by structural induction on the type of gates. A consequence of the proof is that all intermediate shares have been computed by ϕ_i. In particular, that they have to follow the communication pattern of party i solely depending on party $i+1$'s shares. Well-formedness and output share consistency are trivially shown by the original correctness proof, which proves the three criteria for `verify` to succeed.

3-special soundness follows from the security of the additive secret sharing scheme that is used by `decompose` and is given by the original `3-special soundness` proof of the Σ-Protocol.

The proof of 2-privacy carries over directly. Finally, losslessness is trivial. ■

We conclude that ZKBoo fits our general framework.

V. From Decomposition to Σ-Protocol

In this section, we show an example of a black-box construction of a Σ-protocol from the decomposition notion presented in Section IV.

We focus on one of the simplest constructions based on the Σ-protocol by Giacomelli et al. [3]. As we will discuss in Section VI, this construction forms the basis for a family of secure transformations. Note that we obtain a stronger result than Giacomelli et al.: Our construction works for *any* secure decomposition. Moreover, we add the `compress` function to compress views to capture a greater variety of decompositions.

A. Example: ZKBoo Σ-Protocol

Let π be a secure decomposition and Com be a secure commitment scheme. The transformation into a Σ-Protocol is shown in Fig. 4. All references to the internal structure of the decomposition, or even the circuit, are removed. This stands in contrast to Figure. 3.

For the sake of completeness, we will briefly outline the security of this protocol.

Lemma 2: Let π be a secure (k,d)-decomposition, and Com a secure commitment scheme. Then the protocol described in Fig. 4 is a secure Σ-protocol.

Proof:

Completeness: *Verifiability* of the underlying decomposition implies that $\texttt{verify}(\phi, z, ys) \wedge \texttt{rec}(ys) = y$ will always result in true. Verification of commitments, due to our definition of lossless compression, is proven with no additional assumptions on the structure of ws. Since π is a projection, it is easy to determine which parties views are part of the projected views. Hence, for all parties $i \in z, ws[i] = z[i]$. This reduces the verification of commitments to $\texttt{cverify}(\texttt{com}(ws[i]), ws[i]))$ which is ensured by correctness of the commitment scheme. **k-Special Soundness:** Given k verifying transcripts of the Σ-Protocol we can extract k runs of the decomposition, each revealing d views. Because all Σ-Protocol transcripts are computed from distinct challenges, the underlying decomposition must also have been computed from distinct challenges. Moreover, since the randomness and input to the decomposition protocols are fixed, all k runs contain the same shares. In particular, this is also the case for the d projected views. Finally, the binding property of the commitment scheme ensures that all k decomposition views where computed from the same randomness and input.

The consistency of revealed views concludes the proof of k-Special Soundness by application of the soundness property of the decomposition. **Special Honest-Verifier Zero-Knowledge:** Special honest-verifier zero-knowledge is a direct consequence of d-privacy in combination with the hiding property of the commitment scheme which allows simulating commitments. Privacy of the decomposition gives a simulator capable of simulating all output shares of the decomposition and projected views ws', such that $\forall i \in ws', ws'[i] = ws[i]$. Here ws are the views computed by an honest decomposition. For each index $j \notin ws'$, the view $ws[j]$ will not be used by `verify`, nor by `cverify`. Since the views with no simulated counterpart are never accessed, the hiding property of the commitment schemes ensures indistinguishability. ■

VI. Further MPC-in-the-Head Protocols

Numerous implementations of the MPC-in-the-head paradigm for zero-knowledge exist, many of them are optimizations of ZKBoo. In this section, we will briefly discuss how they fit within our definition of a decomposition and how the corresponding transformation to a Σ-protocol change. Note that we consider ZKBoo as the base protocol and explain how the differences of the alternative protocols fit within our framework. This section should not be read as a formal analysis of how the protocols fit within our framework. In particular, the claims in this sections have not been formalised in EasyCrypt. Rather, we aim show how our framework is a natural reformulation of the existing work, which can be transformed without significant changes to the protocol nor proofs.

A. ZKB++

The first protocol is ZKB++ [13]. It offers numerous optimisations for reducing the size of the messages in the Σ-Protocol. Like in ZKBoo, the underlying MPC protocol is kept as a three-party protocol with 2-Privacy. The `compress` functions and the randomness space are optimized. Instead of sampling a long random string at the beginning, the protocol starts by sampling a short seed and proceeds by expanding it into a long pseudo-random one. View compression works as follows: Given a view, the input share and the random seed used to generate all further randomness is projected out. Since all randomness is fixed by the seed it is possible to recompute all shares of the views given the input share. The remaining algorithms for computing the decomposition and verification remain unchanged.

B. KKW

Another optimisation vector was explored by Katz, Kolesnikov and Wang [9]. They replace the traditional MPC protocol with one with preprocessing. This approach splits the MPC protocol into an input-independent offline phase and an online phase where parties use their respective inputs. Essentially, correlated randomness [14] is generated during the offline phase and used in the online phase. The main observation is that since the offline phase is input-independent,

$$\begin{array}{ccl}
\underline{\mathsf{Prover}}(\phi, x, y) & & \underline{\mathsf{Verifier}}(\phi, y) \\
rs \leftarrow_R \mathcal{R} & & \\
ws \leftarrow \texttt{decompose}(\phi, x, rs) & & \\
cs \leftarrow \texttt{map com}\ ws & & \\
ys \leftarrow \texttt{map out}\ rs & & \\
a \leftarrow (cs, ys) & & \\
 & \xrightarrow{a} & \\
 & & e \leftarrow_R \mathcal{C} \\
 & \xleftarrow{e} & \\
z \leftarrow \texttt{map compress}\ \pi(ws, e) & & \\
 & & \texttt{verify}(\phi, map\ \texttt{uncompress}\ z, ys) \\
 & \xrightarrow{z} & \land\ \texttt{rec}(ys) = y \\
 & & \land\ \forall i \in z\colon \texttt{cverify}((\texttt{decompose}\ z[i]), cs[i])
\end{array}$$

Fig. 4: ZKBoo Σ-protocol construction based on secure decomposition.

revealing it completely does not compromise input privacy. Of course, such a revealed offline phase cannot be used in an online phase. So, the work employs a trick to use the repetition of Σ-protocol executions in their favor. Instead of repeating the protocol execution multiple times to reduce the soundness error, like ZKBoo, KKW directly runs m copies of the MPC protocol. The correct execution of the offline phase is then verified via a cut-and-choose approach, i.e. some of the offline phase instances are completely revealed. For the remaining instances, the online phase can then be verified following the ZKBoo template, where the verifier requests the opening of a subset of party views for each instance.

In our terminology, we let $\mathcal{R}$ be the universe of all sets of size m of preprocessed data from the protocol. `decompose` then executes the online phase for each set of provided preprocessed data. `decompose` returns the input of each party, masked under the preprocessing. The challenge set $\mathcal{C}$ is then all tuples of challenges to open a subset of the preprocessing, and challenges to open all but an individual party from the MPC protocols. `compress` selects the subset of runs chosen in the challenge and reveals all preprocessing. For the remaining runs, all preprocessing and views are sent, barring the view of party p.

Since this protocol requires the prover to execute multiple decomposition protocols with the same secret input, but with different initial randomness, the authors added optimisations not only to the MPC protocol but also to the Σ-protocol construction. First, all randomness (preprocessing) is committed to. Then, instead of sending all commitments to the verifier a hash of all preprocessing concatenated is computed. Moreover, all messages of the online phase are concatenated and hashed. The hash of these two hash values is then sent to the verifier. When the prover responds to a challenge, `compress` is used to send the preprocessing and online phase. Additionally, the commitments to the preprocessing of the unrevealed party are sent. `Uncompress` is the identity.

To verify an execution of the Σ-Protocol, the verifier first ensures that the offline and online phase are executed correctly by calling `verify`. Next, the verifier commits to all preprocessing revealed, concatenated it with the commitment of the unrevealed party (when applicable), and computes the hash. The verifier then runs `decompose`, and commits to all messages of the online phase. Finally, the two hash values are hashed again and compared to the value sent by the prover.

C. Picnic

The zero-knowledge protocol underlying the Picnic signature scheme [13] is a combination of the optimizations described above and hence fits nicely within our approach.

D. SNI-in-the-head

Seker et al. [15] showed ZKBoo to be susceptible to probing attacks on the exposed views of the decomposition. To mitigate this attack, the authors then proposed a change to the protocol, in particular how multiplication gates are evaluated. The entire extension conforms to our definitions since only the internal implementation of the `decompose` function is changed compared to ZKBoo. Since the attack is only on the exposed views of the decomposition, the Σ-Protocol does not need to change.

E. BBQ and Banquet

BBQ [6] and Banquet [7] continue the line of optimizations of ZKB++ and KKW and adapt their approach to work with the AES block cipher as the function for the relation to be proved, i.e. the public statement is an AES ciphertext. Using AES is desirable as it is a well-studied and standardized cipher. BBQ uses an MPC protocol in the preprocessing model and can thus be expressed similarly to the KKW protocol. Banquet observes that it is sufficient to compute the verification circuit for correct AES evaluation instead of computing the AES evaluation itself. This change does not affect the applicability of our security notion. Banquet further shows how to improve efficiency by removing the preprocessing again and uses an MPC protocol specifically tailored to evaluate the AES evaluation verification.

VII. EasyCrypt Formalization

In this section, we present how we checked our security proof of ZKBoo (Section III-B) in EasyCrypt. The formalization consists of several parts: We formalize our decomposition notion, introduced in Section IV-A and Σ-protocols. Moreover, we implement our version of the ZKBoo decomposition from Section IV-B and prove it to be a secure decomposition. Finally, we implement and prove the security of a Σ-protocol based on *any* secure decomposition to obtain a complete machine-checked security proof of ZKBoo. In this section, we consider a Σ-protocol for a fixed relation R.

A. *EasyCrypt*

EasyCrypt [10] is a proof assistant designed to capture the code-based game-playing approach to cryptographic proofs [16]. In EasyCrypt, protocols are modelled as probabilistic programs. The tool provides an ambient higher-order logic and an embedded probabilistic relational Hoare logic to reason about a probabilistic `while` language. It offers powerful automation through its interaction with SMT solvers. Proving security of a cryptographic protocol proceeds by proving a series of game transformations. Each transformation either moves a procedure call or substitutes one. This reduction is captured in the relational Hoare logic. Additionally, EasyCrypt has support for defining abstract (ML-style) modules. With abstract modules, one can formulate security specification by quantifying over all possible implementations of a module. This makes black-box style security proofs possible. In such proofs, one only relies on abstract security notions as opposed to concrete implementation details of the protocol.

B. *Σ-Protocol*

We start by explaining the target of our formalization: Σ-protocols. As is common in EasyCrypt, we model this primitive as an abstract module. Similar to the work of Butler et al. in CryptHOL [17], we choose four procedures corresponding to the generation of the three messages exchanged and the final verification step. We generalize their security definitions from 2-Special Soundness to s-Special Soundness. The security properties are then expressed as follows:

- Completeness:

$$\begin{aligned}&\forall h, x, e\colon \mathbf{R}\ h\ x\\ &\quad\implies \Pr[\text{completeness\_game}(h, x, e) = \text{true}] = 1.\end{aligned}$$

- s-Special Soundness:

$$\forall h, x\colon \mathbf{R}\ h\ x \implies \text{real}(h, x, e) \sim \text{ideal}(h, e).$$

- Special Honest-Verifier Zero-Knowledge:

$$\begin{aligned}&\forall h, a, es, vs\colon\\ &\quad(\forall i, 0 \leq i < |es|\colon\\ &\qquad\qquad \Pr[\texttt{sverify}(h, a, es[i], vs[i])] = 1)\\ &\quad\wedge |es| = |vs| \wedge (\forall (e, e') \in es\colon e \neq e')\\ &\quad\implies \Pr[\text{soudness\_game}(h, a, es, vs)] = 1.\end{aligned}$$

The programs used to express game-based security can be seen in Figure 5.

C. *Commitments*

To implement the Σ-protocol we are interested in, we need two components: a commitment scheme and a decomposition. The commitment scheme we use is adapted from Butler et al. [17] and Metere and Dong [18] by changing some game-based definitions to ones in relational Hoare logic. This affects the hiding property, which is more conveniently stated directly as a property of the output distribution of `com`. Again, we formalize the commitment scheme as an abstract module with procedures for the different algorithms according to Def. II-A and the security properties as:

- Correctness:

$$\forall m\colon (c, r) \leftarrow \texttt{com}(c) \implies \Pr[\texttt{cverify}(m, c, r)] = 1.$$

- Hiding:

$$\forall m, m'\colon \texttt{com}(m) \sim \texttt{com}(m').$$

- Binding: $\forall c, m, m', r, r'$:

$$\Pr[\texttt{cverify}(m, c, r) \wedge \texttt{cverify}(m', c, r')] = \epsilon.$$

D. *Decomposition*

The next part is the heart of our formalization: our decomposition from Section IV.

1) Circuits and Views: First, we choose representations for both the circuit and the state of each party. To deal with circuit evaluation, we need a method for associating gates and intermediate computations. This is similar to MPC protocols. We chose to represent both our circuit and views as lists, as this gives us a one-to-one correspondence between gates and shares: the intermediate value for circuit[i] can be found at view[i]. Moreover, lists allow convenient induction proofs.

To facilitate projections of the views we define an index mapping. With this the projection of the views is itself a new list. To reason about the equality between views and projected views we require that $ws[\texttt{proj\_mapping}(i, e)] = \pi(ws, e)[i]$ for all indices i in the projected views. We define the indices of the projection as the views not removed by the projection. For the case of ZKBoo this is view e and $e + 1$.

Throughout this section we use the notation $i \in \pi(ws, e)$ to denote if the view with index i is not removed by the projection.

```
$completeness\_game(h, x, e) =$
    $a \leftarrow \texttt{scommit}(h, x);$
    $z \leftarrow \texttt{sresponse}(h, x, a, e);$
    $\textbf{return } \texttt{sverify}(h, a, e, z);$

$real(h, x, e) =$
    $a \leftarrow \texttt{scommit}(h, x);$
    $z \leftarrow \texttt{sresponse}(h, x, a, e);$
    $\textbf{return } (a, e, z);$

$ideal(h, e) =$
    $(a, z) \leftarrow S(h, e);$
    $\textbf{return } (a, e, z);$

$soundness\_game(h, a, es, zs) =$
    $x' \leftarrow \texttt{extractor}(h, a, es, za);$
    $\textbf{return R } h\ w\ x'$
```

Fig. 5: Σ-Protocol games

2) Security: The security properties are stated in (relational) Hoare logic.

- Verifiability:

$$\begin{aligned}&\forall(\phi : Circuit)(e \in \mathcal{C})(x : Input):\\ &\quad \texttt{valid\_circuit}(\phi) \implies\\ &\quad \Pr[\text{verifiability\_game}(\phi, x, e) = \text{true}] = 1.\end{aligned}$$

- d-Privacy:

$$\begin{aligned}&\forall(\phi : Circuit)(e \in \mathcal{C})(x : Input):\\ &\quad \texttt{real}(\phi, x, e) \sim \texttt{simulator}(\phi, \phi(x), e),\end{aligned}$$

  where real and simulator are defined in definition 6.
- s-Special Soundness:

$$\begin{aligned}&\forall\phi, (es \in \textbf{list } \mathcal{C}), (vs : \textbf{list } view), (ys : \textbf{list } shares):\\ &\quad (\forall i, 0 \leq i < n:\\ &\qquad \Pr[\texttt{verify}(\phi, es[i]), vs[i], ys = true] = 1.)\\ &\quad \wedge |vs| = |es| \wedge \forall(e, e') \in es\colon e \neq e' \wedge |ys| = n\\ &\quad \wedge \texttt{valid\_circuit}(\phi) \wedge \texttt{fully\_consistent}(vs, es)\\ &\quad \implies \Pr\begin{bmatrix}c(\texttt{witness\_extractor}(\phi, \texttt{vs}, \texttt{es})) =\\ \texttt{rec}\ (\texttt{map out}\ ys)\end{bmatrix} = 1.\end{aligned}$$

- Lossleness of compression:
  $\forall(w : View), \texttt{uncompress}(\texttt{compress}(w)) = w.$

The third property uses a helper predicate $\texttt{fully\_consistent}(\{vs_1, \ldots, vs_k\}, \{es_1, \ldots es_k\})$.

A collection of lists of views with respective challenges are fully consistent if the view of the party constrained within two different lists of views vs_a, vs_b are equivalent.

E. ZKBoo Decomposition

With the primitives in place, we can now describe our implementation of ZKBoo and its security proofs.

1) Computation and "communication": The implementation of most procedures of the decomposition is straightforward, the only part that requires thought is decompose. Here we use the gate computation function ϕ from the original ZKBoo paper. While we have removed it from the decomposition notion itself, it is useful in the implementation. We thus fix a procedure

$$\texttt{compute} : \textbf{list view} \times \textbf{gate} \rightarrow \textbf{list share}$$

that updates the views of all parties for gate **gate**. This updating of all shares simultaneously models the communication pattern. All present and past shares are available to the parties, but shares for future gates are unavailable.

2) Randomness sampling: When implementing a probabilistic program there are two ways to sample randomness: lazily and eagerly. Both are equivalent, and both are possible in EasyCrypt. *Eager sampling* samples all randomness at the start of the execution. When a new random value is needed, the next unused value is used. In the case of ZKBoo, that means sampling randomness outside of the decompose procedure. This is necessary for the construction of a Σ-protocol, as that protocol needs some control over the random choices. *Lazy sampling*, on the other hand, samples randomness at the moment it is needed in the protocol. This has the advantage that it allows one to reason about random choices locally. When proving a relational statement, one often wants to relate random choices in two programs via a coupling. This is easier with lazy sampling. For this reason, we define two versions of decompose, one that takes all randomness as input and one that samples internally, and prove them equivalent. The former is more convenient to describe the construction itself, while the latter simplifies the security proof.

3) Security: We prove *verifiability* by showing that the views produced by decompose are computed by the procedure outlined in Section III-B and that they reconstruct to the value of circuit evaluation.

This is achieved by induction on the structure of the circuit. We prove that for every gate computed by the decompositions, if the all previous shares has been computed by the gate evaluation function ϕ_i^j and all previous rounds reconstruct to the same value as the circuit. Then the same holds after computing the next gate.

We can trivially instantiate this fact in the proof of verifiability, since the empty list of shares follows the gate evaluation function for all gates computed so far.

With this in mind, it is immediate that verify always succeeds after π. Since if the invariant holds for all views, it naturally also holds for a subset of the views.

Privacy is proven using a relational statement. For any valid circuit, we show that view e and $e{+}1$ are identically distributed to the two simulated views. By induction on the structure of the circuit, similar to the method used for verifiability, we show that any gate can be simulated. We thus prove, that for any gate simulated, if all shares simulated so far for parties e and $e + 1$ are indistinguishable from the shares of parties e and $e + 1$ computed by the decomposition. Then it also holds after simulating an additional gate.

To facilitate the proof, we rewrite the procedures to use lazy sampling. In this way, we can easily manipulate the random shares in both the simulator and decomposition to make the computed shares indistinguishable. Finally, we reuse the proof from verifiability that the views reconstruct to circuit evaluation. This fixes the output share of the party that is not simulated to be the simulated output value subtracted from the circuit evaluation.

Note that we chose to formalize the simulation-based privacy definition here instead of an equivalent notion of probabilistic non-interference.[3] Non-interference-based privacy definitions for MPC protocols proved hugely successful for the verification of masking scheme privacy in recent years [19]–[21] and were previously used to study the security of secret-sharing-based MPC protocols in EasyCrypt [22]. Using the simulation-based definition, however, means that we stay closer to the cryptographic literature and hopefully make our work more accessible to the cryptographic community. Looking ahead to Section VII-F, the simulation-based approach will moreover enable us to directly construct the zero-knowledge simulator from the MPC privacy simulator, just like one would in a pen-and-paper proof.

To prove k*-Special Soundness* we use `fully_consistent` to derive knowledge of each view in the decomposition. Moreover, the assumption that all revealed views verify allow us to derive that all gates of all views were computed as defined by the decomposition. To show that the input share of the revealed views gives us the secret input for the circuit evaluation, we run the decomposition again. By induction on the number of computed gates, starting from out guess at the secret input, we conclude that our constructed input yields the desired output. Moreover, every gate in the circuit reconstruct to the desired intermediate value. By the reconstruction property proven during verifiability, we can conclude that our guess at the input leads to the correct reconstructed output. This output is equal to the output of circuit evaluation.

4) Proof structure: Both the proof of verifiability and privacy utilises induction on the structure of the circuit to reason about intermediate computations. In particular, the proofs consists of two parts. First, evaluating a single gate maintains a invariant on the values computed. Second, that the invariant is then maintained for the entire circuit.

This structure can facilitate proving verifiability and privacy for other decompositions. For all decomposition-style protocols discussed in Section VI the invariant for both verifiability and privacy remains the same. The only adjustments needed are: Changing the proof of the entire circuit to account for a concrete bounds on n and d. Second, a new proof for the case of a single gate will have to be given.

The first change, our development can be reused with minimal adjustments. For the latter, a new proof will have to be provided. Here we imagine our general definition of a circuit can be used to guide this process.

F. Transformation to Σ-protocol

Finally, we arrive at the Σ-protocol that is our main interest. Due to the security definitions of decompositions (Def. 4), the transformation is black-box and can be constructed independently of implementation details. For our EasyCrypt formalization, this means that the construction is parameterized by an arbitrary decomposition function. We can then instantiate it with the ZKBoo decomposition described above and obtain the ZKBoo protocol.

Let $\mathbf{R}\ (\phi, y)\ x \iff \phi(x) = y$ be the relation of the Σ-protocol. The procedure implementations are seen in Figure 6.

1) Security:

Lemma 3 (Completeness): If the underlying decomposition satisfies verifiability and the commitment scheme is correct, then

$$\begin{aligned}&\forall(\phi : Circuit)(e \in \mathcal{C})(x : Input):\\ &\qquad \mathbf{R}\ h\ x \implies \Pr[\text{Completeness}(\phi, x, e) = \text{true}] = 1.\end{aligned}$$

To prove Completeness, we consider the decomposition and commitment scheme parts separately. By applying verifiability of the decomposition, we see that the verification check will pass, because the views originate from a call to `decompose`. For the commitment scheme, we first use losslessness of the decomposition to derive that the views considered by the verifier are, in fact, identical to the ones produced by the prover. We then apply correctness of the commitment scheme to conclude that the commitments always verify.

Lemma 4 (SHVZK): If the underlying decomposition is d-Private, for any d, and the commitment scheme is perfectly hiding, then

$$\begin{gathered}\forall h, (e \in \mathcal{C}), x:\\ \mathbf{R}\ h\ x \implies real(h, x, e) \sim ideal(h, e).\end{gathered}$$

Where the simulator is defined as:

$$\begin{aligned}&\texttt{simulator}((\phi, y), e) = \{\\ &\quad (z', ys) \leftarrow S_e(\phi, y);\\ &\quad ws' \leftarrow \texttt{map uncompress}\ z';\\ &\quad cs \leftarrow \texttt{map}\left(\begin{aligned}&\lambda i : \texttt{if}\ i \in ws'\\ &\qquad \texttt{then com}(ws'[i])\\ &\qquad \texttt{else com}([\,])\end{aligned}\right)[0..N];\\ &\quad \mathbf{return}\ (ys, cs)\}\end{aligned}$$

In proving Special Honest-Verifier Zero-Knowledge, we first use d-Privacy of the decomposition to show the simulated views revealed by `compress` under challenge e are indistinguishable from the real views. The indistinguishability also implies that both `verify` and `cverify` will succeed since their inputs are indistinguishable from the honestly generated inputs, which are known to succeed. Finally, we use the hiding property of the commitment scheme to conclude that commitments to empty lists are indistinguishable from the commitments to the unrevealed views of the decomposition.

[3] Intuitively, an MPC protocol for function f has non-interference-based privacy if the execution of the protocol on different two sets of inputs $x_1, \dots, x_n$ and $x'_1, \dots, x'_n$ such that $f(x_1, \dots, x_n) = f(x'_1, \dots, x'_n)$ leads to identically distributed views.

```
$\texttt{scommit}((\phi, y), x) = \{$
    $ks \leftarrow \mathcal{R};$
    $ws \leftarrow \texttt{decompose}(\phi, x, ks);$
    $(cs, rs) \leftarrow \texttt{map com}\ ws;$
    $ys \leftarrow \texttt{map out}\ ws;$
    return $(ys, cs);$
$\}$
```

```
$\texttt{sresponse}((\phi, y), (cs, ys), e) = \{$
    $ws' \leftarrow \texttt{map compress}\ \pi(ws, e);$
    return $(ws', \pi(rs, e));$
$\}$
```

```
$\texttt{sverify}((\phi, y), (cs, ys), e, z) = \{$
    $(ws', rs) \leftarrow z;$
    $ws \leftarrow \texttt{map uncompress}\ ws';$
    $v \leftarrow \forall i \in ws\colon \texttt{cverify}(ws[i], cs[\texttt{proj\_mapping}(i, e)], rs[i]);$
    return $v \wedge \texttt{verify}(\phi, z, e, ys);$
$\}$
```

Fig. 6: Σ-Protocol transformation procedures

Lemma 5 (s-Special Soundness): If the underlying decomposition has k-Special Soundness and the commitment scheme is binding with probability $1 - \epsilon$, then

$$\begin{aligned}
&\forall(\phi : \mathit{Circuit}), (es \in \mathcal{C}), y, a, es, zs\colon \\
&\quad (\forall(e, e') \in es\colon e \neq e') \\
&\wedge\ |es| = |vs| = s \wedge \texttt{valid\_circuit}(c) \\
&\wedge\ (\exists(a \in es, b \in es, i)\colon a \neq b \wedge i \in vs[a] \wedge i \in vs[b]) \\
&\wedge\ (\forall i, i < |es|\colon \Pr[\texttt{sverify}(c, y, a, es[i], vs[i]) = \mathit{true}] = 1) \\
&\qquad \Longrightarrow \Pr[\text{soundness\_game}((\phi, y), a, e, z) = \mathit{true}] = (1 - \epsilon).
\end{aligned}$$

From k-Special Soundness of the decomposition, it follows that we can extract a valid witness for the relation. The assumptions of s-Special Soundness therefore implies the assumptions of k-Special Soundness from the decomposition. Concretely, this is achieved by proving:

$$\begin{aligned}
&(\forall i, i < |es|\colon \Pr[\texttt{sverify}(\phi, y, a, es[i], vs[i]) = \text{true}] = 1) \\
&\Longrightarrow \texttt{fully\_consistent}(\texttt{vs}, \texttt{es}).
\end{aligned}$$

To show this, we use the binding property of the commitment scheme. We assume that we are given enough responses, such that at least two responses will overlap on at least one view. Because of this overlap, it follows from the binding property that the two different openings are equivalent.

From the overlap and the proof of equivalence, we derive that the responses are fully consistent.

VIII. Related Work

We compare to existing work on formal verification relating to zero-knowledge protocols.

a) Computational Analysis: One approach is to formalize security proofs of zero-knowledge protocols, which is also the focus of this work. Previous work in this direction include the work of Barthe et at. [23] and Butler, Aspinall and Gascón [17] who focus on formalizing Σ-protocols in CryptHOL [24] and Certicrypt [25], respectively. ZKCrypt by Almeida et al. [26] automatically generates CertiCrypt proofs of the resulting protocols. These approaches have in common that they focus on simpler algebraic protocols, like proving knowledge of pre-images under group homomorphisms. This limits usability to problems that exhibit this simpler algebraic structure. The present work formalizes more sophisticated protocols in which security is reduced to the security of complex building blocks like MPC protocols.

Other works have tried to bridge the gap between formalisation and implementation of zero-knowledge by constructing certified compliers. The work by Almeida et at. [27] develops a specification language for zero-knowledge relations accompanied by a compiler translating the specification to a Σ-Protocol implementation. Notably, the compiler construct a specification for the Σ-Protocol, which is proven secure, before it is complied to the implementation language. The automatic security proofs are only possible due to the special restrictions imposed in their zero-knowledge relations. Namely, all relations expressed are either the pre-image of a homomorphism or the composition of multiple pre-images under (potentially) different homomorphism. Our work does not offer a compiler and follows closer in spirit to the work of Barthe et at., yet we provide a formalisation of a more generalised construction that does not require any special properties of the relation to be proved other than being expressible as a circuit.

Finally, a concurrent independent work by Almeida et al. [28] also studied the security of MPC-in-the-head zero-knowledge protocols in EasyCrypt. The two works differ though in focus and guarantees: Almeida et al. formalized the IKOS approach [2] directly and instantiated it with the general-purpose BGW MPC protocol [29]. The work of Almeida et al. furthermore uses code extraction to obtain an executable OCaml implementation. Our work on the other hand chooses the ZKBoo protocol, a special-purpose construction designed for practical efficiency when combined with the MPC-in-the-head paradigm, as starting point for the formalization and generalizes it to further optimizations. We thus obtain the first formally verified security proof of a practically efficient MPC-in-the-head zero-knowledge protocol, and our work forms the basis for the verification of further optimizations of the ZKBoo approach.

The zero-knowledge protocols that we study use secret-sharing-based MPC with passive security as a building block. This type of MPC protocol has been formalized previously by Butler, Aspinall and Gascón [30] and Haagh et al. [22]. Our MPC protocol formalization is close in spirit to the passive security construction of Haagh et al., but differs in that we directly formalize a simulation-based security notion which is more familiar to cryptographers than the non-interference used there.

b) Symbolic Analysis: An orthogonal line of work studies the symbolic security of protocols that use zero-knowledge protocols as primitives [31], [32]. In this setting, the zero-knowledge proofs themselves are treated as abstract objects that can be manipulated according to fixed rules modelled as equational theory. Symbolic security of a protocol then rules out any attack that follows only those allowed manipulations. This approach cannot capture the security of a concrete zero-knowledge protocol, but only of another protocol that uses it.

IX. Discussion and Future Work

The present work shows how formal verification cannot only recreate existing proofs but also foster a deeper understanding of the object in question. In our case, we set out to formalize the ZKBoo security proof and found out that what looked like a modular proof structure was not as modular as it could be. Obvious future work includes extending our efforts to more efficient protocols following the MPC-in-the-head paradigm, especially, if any of them becomes standardized. As mentioned, Picnic was recently announced as an 'alternate' in the third round of the NIST post-quantum cryptography standardization competition. The reason Picnic is an alternate, and not a candidate, is that several improvements were published after the submission of Picnic. Once the line of research converges to one, or more, efficient constructions ready for standardization, we expect our work to form the basis of further formal verification efforts. One could even consider connecting our work with an actual implementation. Simultaneously, the structures that we identified in this work enhance the understanding of the MPC-in-the-head paradigm and can provide insights into possibilities for further optimization and constructions.

The Picnic protocol, more so than being an efficient MPC-in-the-head protocol, also serves as the fundamental building block for a post-quantum secure signature scheme. With this work, we now have a better understanding of the MPC-in-the-head construction. This could form the basis for reasoning about the signature scheme construction.

Last, this work could be extended to construct a zero-knowledge (rather than honest verifier zero-knowledge) protocol through the Fiat-Shamir transformation [4]. This transformation is well-studied in cryptographic literature, but currently, no EasyCrypt formalisation of it exists.

X. Conclusion

We initiated the formal analysis of zero-knowledge protocols following the MPC-in-the-head paradigm. Based on the observation that existing constructions are black-box in the MPC protocol they use while their security analysis is not, we proposed a new security notion for these MPC protocols. This modular security proof then enabled us to develop a machine-checked security proof of the ZKBoo protocol in EasyCrypt as an example of this protocol class.

Additionally, we have defined a methodology for reasoning about our security properties for circuit-based MPC protocol. With this proof methodology, we allow for parts of our work to be directly re-used in further formalisation of MPC-in-the-head approach.

Acknowledgments: The authors would like to thank Claudio Orlandi for helpful discussions. This work was partially supported by AFOSR grant *Homotopy type theory and probabilistic computation* (12595060) and the Concordium Blockchain Research Center at Aarhus University. The second author was supported by the Danish Independent Research Council under Grant-ID DFF-8021-00366B (BETHE).

References

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems (extended abstract)," in *17th ACM STOC*. ACM Press, May 1985, pp. 291–304.

[2] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation," in *39th ACM STOC*, D. S. Johnson and U. Feige, Eds. ACM Press, Jun. 2007, pp. 21–30.

[3] I. Giacomelli, J. Madsen, and C. Orlandi, "ZKBoo: Faster zero-knowledge for Boolean circuits," in *USENIX Security 2016*, T. Holz and S. Savage, Eds. USENIX Association, Aug. 2016, pp. 1069–1083.

[4] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO'86*, ser. LNCS, A. M. Odlyzko, Ed., vol. 263. Springer, Heidelberg, Aug. 1987, pp. 186–194.

[5] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales, "Picnic," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[6] C. de Saint Guilhem, L. De Meyer, E. Orsini, and N. P. Smart, "BBQ: Using AES in picnic signatures," in *SAC 2019*, ser. LNCS, K. G. Paterson and D. Stebila, Eds., vol. 11959. Springer, Heidelberg, Aug. 2019, pp. 669–692.

[7] C. Baum, C. Delpech de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha, "Banquet: Short and fast signatures from AES," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 68, 2021. [Online]. Available: https://eprint.iacr.org/2021/068

[8] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," Cryptology ePrint Archive, Report 2017/279, 2017, http://eprint.iacr.org/2017/279.

[9] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *ACM CCS 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM Press, Oct. 2018, pp. 525–537.

[10] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella Béguelin, "Computer-aided security proofs for the working cryptographer," in *CRYPTO 2011*, ser. LNCS, P. Rogaway, Ed., vol. 6841. Springer, Heidelberg, Aug. 2011, pp. 71–90.

[11] I. Damgaard, "On Σ-protocols," lecture notes, Aarhus University, 2011.

[12] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *EUROCRYPT'96*, ser. LNCS, U. M. Maurer, Ed., vol. 1070. Springer, Heidelberg, May 1996, pp. 387–398.

[13] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," in *ACM CCS 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 1825–1842.

[14] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky, "On the power of correlated randomness in secure computation," in *TCC 2013*, ser. LNCS, A. Sahai, Ed., vol. 7785. Springer, Heidelberg, Mar. 2013, pp. 600–620.

[15] O. Seker, S. Berndt, L. Wilke, and T. Eisenbarth, "SNI-in-the-head: Protecting MPC-in-the-head protocols against side-channel analysis," in *ACM CCS 20*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM Press, Nov. 2020, pp. 1033–1049.

[16] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *EUROCRYPT 2006*, ser. LNCS, S. Vaudenay, Ed., vol. 4004. Springer, Heidelberg, May / Jun. 2006, pp. 409–426.

[17] D. Butler, D. Aspinall, and A. Gascón, "On the formalisation of Σ-protocols and commitment schemes," in *Principles of Security and Trust - 8th International Conference, POST 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, ser. Lecture Notes in Computer Science, F. Nielson and D. Sands, Eds., vol. 11426. Springer, 2019, pp. 175–196.

[18] R. Metere and C. Dong, "Automated cryptographic analysis of the pedersen commitment scheme," *CoRR*, vol. abs/1705.05897, 2017. [Online]. Available: http://arxiv.org/abs/1705.05897

[19] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, and P.-Y. Strub, "Verified proofs of higher-order masking," in *EUROCRYPT 2015, Part I*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, Heidelberg, Apr. 2015, pp. 457–485.

[20] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini, "Strong non-interference and type-directed higher-order masking," in *ACM CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM Press, Oct. 2016, pp. 116–129.

[21] G. Barthe, S. Belaïd, G. Cassiers, P.-A. Fouque, B. Grégoire, and F.-X. Standaert, "maskVerif: Automated verification of higher-order masking in presence of physical defaults," in *ESORICS 2019, Part I*, ser. LNCS, K. Sako, S. Schneider, and P. Y. A. Ryan, Eds., vol. 11735. Springer, Heidelberg, Sep. 2019, pp. 300–318.

[22] H. Haagh, A. Karbyshev, S. Oechsner, B. Spitters, and P.-Y. Strub, "Computer-aided proofs for multiparty computation with active security," in *CSF 2018Computer Security Foundations Symposium*, S. Chong and S. Delaune, Eds. IEEE Computer Society Press, 2018, pp. 119–131.

[23] G. Barthe, D. Hedin, S. Zanella-Béguelin, B. Grégoire, and S. Heraud, "A machine-checked formalization of sigma-protocols," in *CSF 2010Computer Security Foundations Symposium*, A. Myers and M. Backes, Eds. IEEE Computer Society Press, 2010, pp. 246–260.

[24] D. A. Basin, A. Lochbihler, and S. R. Sefidgar, "CryptHOL: Game-based proofs in higher-order logic," *Journal of Cryptology*, vol. 33, no. 2, pp. 494–566, Apr. 2020.

[25] G. Barthe, B. Grégoire, and S. Z. Béguelin, "Formal certification of code-based cryptographic proofs," in *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, Z. Shao and B. C. Pierce, Eds. ACM, 2009, pp. 90–101.

[26] J. B. Almeida, M. Barbosa, E. Bangerter, G. Barthe, S. Krenn, and S. Zanella Béguelin, "Full proof cryptography: verifiable compilation of efficient zero-knowledge protocols," in *ACM CCS 2012*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM Press, Oct. 2012, pp. 488–500.

[27] J. B. Almeida, E. Bangerter, M. Barbosa, S. Krenn, A.-R. Sadeghi, and T. Schneider, "A certifying compiler for zero-knowledge proofs of knowledge based on sigma-protocols," in *ESORICS 2010*, ser. LNCS, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds., vol. 6345. Springer, Heidelberg, Sep. 2010, pp. 151–167.

[28] J. C. B. Almeida, M. Barbosa, K. Eldefrawy, S. Graham-Lengrand, H. Pacheco, and V. Pereira, "Machine-checked ZKP for np-relations: Formally verified security proofs and implementations of mpc-in-the-head," *CoRR*, vol. abs/2104.05516, 2021. [Online]. Available: https://arxiv.org/abs/2104.05516

[29] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *20th ACM STOC*. ACM Press, May 1988, pp. 1–10.

[30] D. Butler, D. Aspinall, and A. Gascón, "How to simulate it in isabelle: Towards formal proof for secure multi-party computation," in *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, ser. Lecture Notes in Computer Science, M. Ayala-Rincón and C. A. Muñoz, Eds., vol. 10499. Springer, 2017, pp. 114–130.

[31] M. Backes and D. Unruh, "Computational soundness of symbolic zero-knowledge proofs against active attackers," in *CSF 2008Computer Security Foundations Symposium*, A. Sabelfeld, Ed. IEEE Computer Society Press, 2008, pp. 255–269.

[32] M. Backes, M. Maffei, and D. Unruh, "Zero-knowledge in the applied Pi-calculus and automated verification of the direct anonymous attestation protocol," in *2008 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2008, pp. 202–215.