Fiat-Shamir Transformation of Multi-Round Interactive Proofs

Thomas Attema[1,3,4,⋆], Serge Fehr[1,3,⋆⋆], and Michael Klooß[2,⋆⋆⋆]

[1] CWI, Cryptology Group, Amsterdam, The Netherlands
[2] KASTEL, Karlsruhe Institute of Technology, Karlsruhe, Germany
[3] Leiden University, Mathematical Institute, Leiden, The Netherlands
[4] TNO, Cyber Security and Robustness, The Hague, The Netherlands

Abstract. The celebrated Fiat-Shamir transformation turns any public-coin interactive proof into an non-interactive one, which inherits the main security properties (in the random oracle model) of the interactive version. While originally considered in the context of 3-move public-coin interactive proofs, i.e., so-called Σ-protocols, it is now applied to multi-round protocols as well. Unfortunately, the security loss for a $(2\mu+1)$-move protocol is, in general, Q^μ, where Q is the number of oracle queries performed by the attacker. In general, this is the best one can hope for, as it is easy to see that this loss applies to the μ-fold sequential repetition of Σ-protocols, but it raises the question whether certain (natural) classes of interactive proofs feature a milder security loss.
In this work, we give positive and negative results on this question. On the positive side, we show that for $(k_1,\ldots,k_\mu)$-special-sound protocols (which cover a broad class of use cases), the knowledge error degrades linearly in Q (instead of Q^μ). On the negative side, we show that for t-fold *parallel* repetitions of typical $(k_1,\ldots,k_\mu)$-special-sound protocols, there is an attack which results in a security loss of about $(Q/\mu)^\mu\mu^{-t}$, assuming for simplicity that t is an integer multiple of μ.

1 Introduction

1.1 Background

The celebrated and broadly used Fiat-Shamir transformation turns any public-coin interactive proof into a *non-interactive* proof, which inherits the main security properties (in the random oracle model) of the interactive version. The rough idea is to replace the random challenges, which are provided by the verifier in the interactive version, by the hash of the current message (concatenated with the message-challenge pairs from previous rounds). By a small adjustment, where also the to-be-signed message is included in the hashes, the transformation turns any public-coin interactive proof into a signature scheme. Indeed, the latter is a commonly used design principle for constructing very efficient signature schemes.

While originally considered in the context of 3-move public-coin interactive proofs, i.e., so-called Σ-protocols, the Fiat-Shamir transformation also applies to *multi-round* protocols. However, a major drawback in the case of multi-round protocols is that, in general, the security loss obtained by applying the Fiat-Shamir transformation grows exponentially with the number of rounds. Concretely, for any $(2\mu+1)$-move interactive proof Π (where we may assume that the prover speaks first and last, so that the number of communication rounds is indeed odd) that admits a cheating probability of at most ϵ, e.g., captured by the (knowledge) soundness error, the Fiat-Shamir-transformed protocol $\mathsf{FS}[\Pi]$ admits a cheating probability of at most $(Q+1)^\mu\epsilon$, where Q denotes the number of random-oracle queries admitted to the dishonest prover. Furthermore, there are (contrived) examples of multi-round protocols Π for which this $(Q+1)^\mu$ security loss is almost tight. For instance, the μ-fold sequential repetition Π of a special-sound Σ-protocol with challenge space $\mathcal{C}$ is ϵ-sound with $\epsilon = \frac{1}{|\mathcal{C}|^\mu}$, while it is easy to see that $\mathsf{FS}[\Pi]$ can be broken with probability $\left(\frac{Q}{\mu}\frac{1}{|\mathcal{C}|}\right)^\mu = \left(\frac{Q}{\mu}\right)^\mu\epsilon$.[1]

⋆ thomas.attema@tno.nl
⋆⋆ serge.fehr@cwi.nl
⋆⋆⋆ michael.klooss@kit.edu

[1] This is clearly a contrived example since the natural construction would be to apply the Fiat-Shamir transformation to the *parallel* repetition of the original Σ-protocol, where no such huge security loss would then occur.

For μ beyond 1 or 2, let alone for non-constant μ (e.g., for Bulletproofs-like protocols [BCC+16,BBB+18]), this is a very unfortunate situation when it comes to choosing concrete security parameters. If one wants to rely on the proven security reduction, one needs to choose the security parameter for Π so large, in order to compensate for the order Q^μ security loss, that the scheme becomes totally unpractical; alternatively, one has to give up on proven security and simply *assume* that the security loss is much milder than what the general bound suggests — indeed, for the protocols one cares about, the *known* attacks do not feature such a large security loss. The latter has become common practice.

This situation gives rise to the following question: *Do there exist natural classes of multi-round public-coin interactive proofs for which the security loss behaves more benign than what the general reduction suggests?* Ideally, the general Q^μ loss appears for contrived examples *only*. So far, the only positive result in that direction is [GT21], which shows an online/straight-line extractor for Bulletproofs and related protocols in the *algebraic group model*. They prove a linear loss of security linear in Q (and linear in n, the statement size).

In this work, we address this question (in the plain random-oracle model), and give both positive and negative answers, as explained in more detail below.

1.2 Our Results

On the positive side, we show that the Fiat-Shamir transformation of any $(k_1, \ldots, k_\mu)$-special-sound interactive proof has a security loss of at most $Q+1$. More concretely, we consider the *knowledge error* κ as the figure of merit, i.e., informally, the maximal probability of a the verifier accepting the proof when the prover does not have a witness for the claimed statement, and we prove the following result. For any $(k_1, \ldots, k_\mu)$-special-sound $(2\mu+1)$-move interactive proof Π with knowledge error κ (which is a known function of $(k_1, \ldots, k_\mu)$), the Fiat-Shamir transformed protocol $\mathsf{FS}[\Pi]$ has a knowledge error at most $(Q+1)\kappa$.

Since in the Fiat-Shamir transformation of a $(2\mu+1)$-move protocol Π, a dishonest prover can simulate any attack against Π, and can try Q/μ times when allowed to do Q queries in total, our new upper bound $(Q+1)\kappa$ is close to the trivial lower bound $Q\kappa/\mu$. Another, less explicit, security measure in the context of knowledge soundness is the run time of the knowledge extractor. Our bound on the knowledge error holds by means of a knowledge extractor that makes an expected number of $K + Q(K-1)$ queries, where $K = k_1 \cdots k_\mu$. This is a natural bound: K is the number of necessary distinct "good" transcripts (which form a certain tree-like structure). The loss of $Q(K-1)$ captures the fact that a prover may finish different proofs, depending on the random oracle answers, and only one out of Q proofs may be useful for extraction, as explained below.

The construction of our knowledge extractor is motivated by the extractor from [ACK21] in the interactive case, but the analysis here in the context of a non-interactive proof is much more involved. We analyze the extractor in an inductive manner, and capture the induction step (and the base case) by means of an abstract experiment. The crucial idea for the analysis (and extractor) is how to deal with accepting transcripts which are not useful.

To see the problem, consider a Σ-protocol, i.e., a 3-move k-special-sound interactive proof, and a semi-honest prover, which knows a witness, and behaves as follows. It prepares, independently, Q first messages $a^1, \ldots, a^Q$ and asks for all hashes $c^i = \mathsf{RO}(a^i)$, and then decides "randomly" (e.g., using a hash over all random oracle answers) which thread to complete, i.e., for which i^* to compute the response z^{i^*} and output the valid proof (a^{i^*}, z^{i^*}). When the extractor then reprograms the random oracle to try to obtain another valid response but now for a different challenge, this affects i^*, and most likely the prover will then use a different thread. Hence, an overhead of Q appears in the run-time.

Perhaps surprisingly, dealing with the knowledge error is relatively simple, even when recursively composing the extractor. However, controlling the run-time is intricate. If the extractor is recursively composed, i.e., it makes calls to a subextractor to obtain a subtree, then a naive construction and analysis gives a blow-up of Q^μ in the run-time. Intuitively, because only $1/Q$ of the subextractor runs produce useful subtrees, i.e., subtrees which extend the current a^{i^*}. The other trees belong to some a^j with $j \neq i^*$ and are thus useless. This overhead of Q then accumulates per challenge (i.e., per subextractor).

The crucial observation that we exploit in order to overcome the above issue is that the very first (accepting) transcript sampled by a subextractor already determines whether a subtree will be useful or not. Thus, if this very first transcript already shows that the subtree will not be useful, there is no need to run the full-fledged subtree extractor, saving precious time.

To formally capture the technical aspects behind the extractor analysis, we consider and analyze an abstract sampling game. The experiment considers a high-dimensional array of balls, each one having one out of two possible colours as well as a pointer to one of the many dimensions, and the goal is to find, by means of a prescribed strategy (which reflects how the extractor proceeds), k balls with the right colour, pointers to the same dimension, and appropriately located, to be defined in detail later on. The technical core of our proof then lies in analyzing certain figures of merit in this abstract experiment: the success probability and a cost function. Defining the cost function naively as the (expected) number of balls that need to be picked is good enough for the analysis of the extractor of a Σ-protocol, but would lead to the old Q^μ blow-up when analyzing the inductively defined extractor in that way. In order to capture the above idea of not running the full-fledged subtree extractor when it can be avoided, we introducing two weight functions and defining the cost function by means of the total weight of the picked balls.

On the negative side, we show that the general exponential security loss of the Fiat-Shamir transformation, when applied to a multi-round protocol, is *not* an artefact of contrived examples, but there exist *natural* protocols that indeed have such an exponential loss. Concretely, we show that the μ-fold parallel repetition Π^t of a typical $(k_1, \ldots, k_\mu)$-special-sound $(2\mu + 1)$-move interactive proof Π features this behavior. For simplicity, let us assume that t and Q are multiples of μ. Then, in more detail, we show that for any typical $\mathbf{k}$-special sound protocol Π there exists a poly-time Q-query prover $\mathcal{P}^*$ against $\mathsf{FS}[\Pi^t]$ that succeeds in making the verifier accept with probability $(Q/\mu)^\mu(\kappa/\mu)^t$ for *any* statement x, where κ is the knowledge error (as well as the soundness error) of Π. Thus, with the claimed probability, $\mathcal{P}^*$ succeeds in making the verifier accept for statements x that are not in the language and/or for which $\mathcal{P}^*$ does not know a witness. Given that κ^t is the soundness error of Π^t (i.e., the soundness error of Π^t as an interactive proof), when setting $t = \mu$ this shows that the soundness error of Π^μ grows with a factor Q^μ when applying the Fiat-Shamir transformation. Recent work on the knowledge error of a parallel repetition [AF21] shows that κ^t is also the knowledge error of Π^t, showing the same exponential loss in the knowledge error of the Fiat-Shamir transformation of a parallel repetition.

1.3 Related Work

Independent Concurrent Work. In independent and to a large extent concurrent work,[2] Wikström [Wik21] achieves a similar positive result on the Fiat-Shamir transformation, using a different approach and different techniques: [Wik21] reduces non-interactive extraction to a form of interactive extraction and then applies a generalized version of [Wik18], while our construction adapts the interactive extractor from [ACK21] and offers a direct analysis. One small difference in the results, which is mainly of theoretical interest, is that our result holds and is meaningful for *any* $Q < N$, where N is the size of the challenge set, whereas [Wik21] requires N to be large.

The Forking Lemma. Security of the Fiat–Shamir transformation of k-special-sound 3-move protocols is widely used for construction of signatures. There, unforgeability is typically proven via a forking lemma [PS96,BN06], which extracts, with probability roughly ϵ^k/Q, a witness from a signature-forging adversary with success probability ϵ, where Q is the number of queries to the random oracle. The loss ϵ^k is due to *strict* polynomial time extraction (and can be decreased, but in general not down to ϵ). Such a k-th power loss in the success probability for a constant k is fine in certain settings, e.g., for proving the security of signature schemes; however, not for proofs of knowledge (which, on the other hand, consider *expected* polynomial time extraction [BL02]). We are not aware of forking lemmas being used in the context of the Fiat-Shamir transformation for *multi-round* interactive proofs, i.e., for $(2\mu + 1)$-move protocols with $\mu > 1$.

[2] When finalizing our write-up, we were informed by Wikström that he derived similar results a few months earlier, subsequently made available online [Wik21].

2 Preliminaries

2.1 Interactive Proofs

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. Following standard conventions, we call $(x; w) \in R$ a statement-witness pair, that is, x is the *statement* and w is a *witness* for x. The set of valid witnesses for a statement x is denoted by $R(x)$, i.e., $R(x) = \{w : (x; w) \in R\}$. A statement that admits a witness is said to be a *true* or *valid* statement; the set of true statements is denoted by L_R, i.e., $L_R = \{x : \exists\, w \text{ s.t. } (x; w) \in R\}$. The relation R is an NP relation if the validity of a witness w can be verified in time polynomial in the size $|x|$ of the statement x. From now on we assume all relations to be NP relations.

In an interactive proof for a relation R, a prover $\mathcal{P}$ aims to convince a verifier $\mathcal{V}$ that a statement x admits a witness, or even that the prover *knows* a witness $w \in R(x)$.

Definition 1 (Interactive Proof). *An* interactive proof $(\mathcal{P}, \mathcal{V})$ *for relation R is an interactive protocol between two probabilistic machines, a prover $\mathcal{P}$ and a polynomial time verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ take as public input a statement x and, additionally, $\mathcal{P}$ takes as private input a witness $w \in R(x)$. The verifier $\mathcal{V}$ either accepts or rejects and its output is denoted as $(\mathcal{P}(w), \mathcal{V})(x)$. Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is* accepting *or* rejecting.

Let us introduce some conventions and additional properties for interactive proof systems. We assume that the prover $\mathcal{P}$ sends the first and the last message in any interactive proof $(\mathcal{P}, \mathcal{V})$. Hence, the number of communication moves $2\mu + 1$ is always odd. We also say $(\mathcal{P}, \mathcal{V})$ is a $(2\mu + 1)$-move protocol. We will refer to *multi-round* protocols as a way of emphasizing that we are not restricting to 3-move protocols.

An interactive proof system $(\mathcal{P}, \mathcal{V})$ is *complete*, if any honest execution, for statement-witness pair $(x; w) \in R$, results in the verifier accepting with high probability. It is *sound* if the verifier rejects false statements, i.e., $x \notin L_R$, with high probability. We do neither require (nor formally define) completeness and soundness, as our main focus is *knowledge soundness*. Intuitively, a protocol is knowledge sound if any (potentially malicious) prover $\mathcal{P}^*$ which convinces the verifier of the truth of a statement x, i.e., $x \in L_R$, must "know" a witness w such that $(x, w) \in R$. Informally, this means that any prover $\mathcal{P}^*$ with $\Pr((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept})$ large enough is able to efficiently compute a witness $w \in R(x)$.

Definition 2 (Knowledge Soundness). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for relation R is* knowledge sound *with knowledge error $\kappa\colon \{0,1\}^* \to [0,1]$ if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a* knowledge extractor*, with the following properties: Given input x and black-box oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, the extractor $\mathcal{E}$ runs in an expected number of steps that is polynomial in $|x|$ (counting queries to $\mathcal{P}^*$ as a single step) and outputs a witness $w \in R(x)$ with probability*

$$\Pr\big((x; \mathcal{E}^{\mathcal{P}^*}(x)) \in R\big) \geq \frac{\epsilon(\mathcal{P}^*, x) - \kappa(x)}{q(|x|)},$$

where $\epsilon(\mathcal{P}^*, x) := \Pr((\mathcal{P}^*, \mathcal{V})(x) = \mathsf{accept})$.

Note that black-box access to $\mathcal{P}^*$ allows $\mathcal{E}$ to "rewind" $\mathcal{P}^*$ to any state.

Remark 1. From linearity of expectation, it follows easily that it is sufficient to consider *deterministic* provers $\mathcal{P}^*$ in Definition 2. Consequently, we will assume all (malicious) provers to be deterministic to simplify our analysis.

An important class of protocols have particularly simple verifiers: Effectively stateless verifiers which send uniformly random challenges to the prover, and run an efficient verification function on the final transcript.

Definition 3 (Public-Coin). *An interactive proof system $(\mathcal{P}, \mathcal{V})$ is* public-coin *if all of $\mathcal{V}$'s random choices are made public. The message $c_i \leftarrow_R \mathcal{C}_i$ of $\mathcal{V}$ in the $2i$-th move is called the i-th* challenge*, and $\mathcal{C}_i$ is the* challenge set*.*

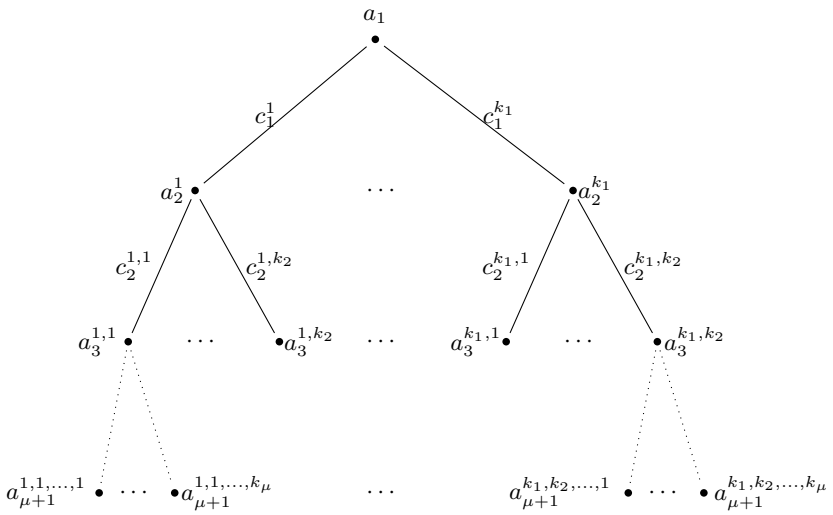

Fig. 1. $(k_1, \ldots, k_\mu)$-tree of transcripts of a $(2\mu+1)$-move public-coin interactive proof [ACK21].

The class of protocols we are interested in are those where knowledge soundness follows from another property, namely *special-soundness*. Special-soundness is often simpler to verify, and many protocols satisfy this notion. Note that we require special-sound protocols to be public-coin.

Definition 4 (k-out-of-N Special-Soundness). *Let $k, N \in \mathbb{N}$. A 3-move public-coin interactive proof $(\mathcal{P}, \mathcal{V})$ for relation R, with challenge set of cardinality $N \geq k$, is k-out-of-N special-sound if there exists a polynomial time algorithm that, on input a statement x and k accepting transcripts $(a, c_1, z_1), \ldots (a, c_k, z_k)$ with common first message a and pairwise distinct challenges $c_1, \ldots, c_k$, outputs a witness $w \in R(x)$. We also say $(\mathcal{P}, \mathcal{V})$ is k-special-sound and, if $k = 2$, it is simply said to be special-sound.*

We refer to a 3-move public-coin interactive proof as a *Σ-protocol*. Note that often a Σ-protocol is required to be (perfectly) complete, special-sound and special honest-verifier zero-knowledge (SHVZK) by definition. However, we do not require a Σ-protocol to have these additional properties.

Definition 5 (Σ-Protocol). *A Σ-protocol is a 3-move public-coin interactive proof.*

In order to generalize k-special-soundness to multi-round protocols we introduce the notion of a tree of transcripts. We follow the definition of [ACK21].

Definition 6 (Tree of Transcripts). *Let $k_1, \ldots, k_\mu \in \mathbb{N}$. A $(k_1, \ldots, k_\mu)$-tree of transcripts for a $(2\mu+1)$-move public-coin interactive proof $(\mathcal{P}, \mathcal{V})$ is a set of $K = \prod_{i=1}^{\mu} k_i$ transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node. For a graphical representation we refer to Figure 1. We refer to the corresponding tree of challenges as a $(k_1, \ldots, k_\mu)$-tree of challenges.*

We will also write $\mathbf{k} = (k_1, \ldots, k_\mu) \in \mathbb{N}^\mu$ and refer to a $\mathbf{k}$-tree of transcripts or a $\mathbf{k}$-tree of challenges.

Definition 7 ($(k_1, \ldots, k_\mu)$-out-of-$(N_1, \ldots, N_\mu)$ Special-Soundness). *Let $k_1, \ldots, k_\mu, N_1, \ldots, N_\mu \in \mathbb{N}$. A $(2\mu+1)$-move public-coin interactive proof $(\mathcal{P}, \mathcal{V})$ for relation R, where $\mathcal{V}$ samples the i-th challenge from a set of cardinality $N_i \geq k_i$ for $1 \leq i \leq \mu$, is $(k_1, \ldots, k_\mu)$-out-of-$(N_1, \ldots, N_\mu)$ special-sound if there exists a polynomial time algorithm that, on input a statement x and a $(k_1, \ldots, k_\mu)$-tree of accepting transcripts outputs a witness $w \in R(x)$. We also say $(\mathcal{P}, \mathcal{V})$ is $(k_1, \ldots, k_\mu)$-special-sound.*

It is well known that, for 3-move protocols, k-special-soundness implies knowledge soundness, but only recently it was shown that more generally, for public-coin $(2\mu+1)$-move protocols, $(k_1,\ldots,k_\mu)$-out-of-$(N_1,\ldots,N_\mu)$ special-soundness tightly implies ordinary and knowledge soundness [ACK21], with soundness/knowledge error

$$\mathrm{Er}(k_1,\ldots,k_\mu;N_1,\ldots,N_\mu) = 1-\prod_{i=1}^{\mu}\frac{N_i-k_i+1}{N_i} = 1-\prod_{i=1}^{\mu}\left(1-\frac{k_i-1}{N_i}\right), \tag{1}$$

which matches the probability that at least one of the random challenges c_i hits a given set S_i of size k_i-1. Note that $\mathrm{Er}(k;N)=(k-1)/N$ and, for all $1\le m\le\mu$,

$$\mathrm{Er}(k_m,\ldots,k_\mu;N_m,\ldots,N_\mu) = 1-\frac{N_m-k_m+1}{N_m}\big(1-\mathrm{Er}(k_{m+1},\ldots,k_\mu;N_{m+1},\ldots,N_\mu)\big), \tag{2}$$

where we define $\mathrm{Er}(\emptyset;\emptyset)=1$. If $N_1=\cdots=N_\mu=N$, i.e., if the verifier samples all μ challenges from a set of size N, we simply write $\mathrm{Er}(k_1,\ldots,k_\mu;N)$, or $\mathrm{Er}(\mathbf{k};N)$ for $\mathbf{k}=(k_1,\ldots,k_\mu)$.

2.2 Non-interactive random oracle proofs (NIROP)

In practice, interactive proofs are not typically used. Instead, transformations are used which turn them into non-interactive proofs in the random oracle model. We define non-interactive random oracle proofs (NIROP) as in [BCS16]. Their definition is a straightforward adaption of (non-)interactive proof systems to the ROM. The same holds for their properties. Every algorithm is augmented by access to a random oracle.

In the *random oracle model*, algorithms have black-box access to an oracle $\mathsf{RO}\colon\{0,1\}^*\to\mathcal{Y}$, called the *random oracle*, which is instantiated by a uniformly random function with domain $\{0,1\}^*$ and codomain $\mathcal{Y}$. For convenience, we let the codomain $\mathcal{Y}$ be an arbitrary finite set, while typically $\mathcal{Y}=\{0,1\}^{2\lambda}$, where λ is the security parameter. Equivalently, RO is instantiated by lazy sampling, i.e., for every bit-string $x\in\{0,1\}^*$, $\mathsf{RO}(x)$ is chosen uniformly at random (and then fixed). To avoid technical difficulties, we limit the domain from $\{0,1\}^*$ to $\{0,1\}^{\le u}$, the finite set of all bitstring of length at most u, for a sufficiently large $u\in\mathbb{N}$. An algorithm $\mathcal{A}^{\mathsf{RO}}$, that is given black-box access to a random oracle, is called a *random-oracle algorithm*. We say that $\mathcal{A}$ is a *Q-query random-oracle algorithm*, if it makes at most Q queries to RO (for any choice of RO).

A natural extension of the random oracle model is when $\mathcal{A}$ is given access to *multiple independent* random oracles $\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu$, possibly with different codomains.[3] The definitions below apply to multiple random oracles in the obvious way.

Definition 8 (Non-Interactive Random Oracle Proof (NIROP)). *A* non-interactive random oracle proof *for relation R and language L_R is a pair $(\mathcal{P},\mathcal{V})$ of (probabilistic) random-oracle algorithms, a prover $\mathcal{P}$ and a polynomial-time verifier $\mathcal{V}$, such that: Given $(x;w)\in R$ and access to a random oracle RO, the prover $\mathcal{P}^{\mathsf{RO}}(x;w)$ outputs a proof π. Given $x\in\{0,1\}^*$, a purported proof π, and access to a (random) oracle RO, the verifier $\mathcal{V}^{\mathsf{RO}}(x,\pi)$ outputs 0 to reject or 1 to accept the proof.*

As for interactive definitions, a NIROP is complete if honestly generated proofs for $(x;w)\in R$ are accepted by $\mathcal{V}$ with high probability. They are sound if it is infeasible to produce an accepting proof for a false statement. In the non-interactive setting, the soundness error, i.e., the success probability of a cheating prover necessarily depends on the number of queries it is allowed to make to the random oracle. The same holds true for knowledge soundness of NIROPs.

Definition 9 (Knowledge Soundness). *A non-interactive random oracle proof $(\mathcal{P},\mathcal{V})$ for relation R is* knowledge sound *with* knowledge error $\kappa\colon\{0,1\}^*\times\mathbb{N}\to[0,1]$ *if there exists a positive polynomial q and an*

[3] In practice, these random oracles will be instantiated by one random oracle $\mathsf{RO}:\{0,1\}^*\to\{0,1\}^{2\lambda}$ using standard techniques for domain separation and sampling algorithms to choose from $\mathcal{C}_i$ using binary coins.

algorithm $\mathcal{E}$, called a knowledge extractor*, with the following properties: The extractor, given input x and oracle access to any (potentially dishonest) Q-query random oracle prover $\mathcal{P}$, runs in an expected number of steps that is polynomial in $(|x|, Q)$ and outputs a witness $w \in R(x)$, and satisfies*

$$\Pr\big((x;w) \in R : w \leftarrow \mathcal{E}^{\mathcal{P}}(x)\big) \geq \frac{\epsilon(\mathcal{P}^*, x) - \kappa(x, Q)}{q(|x|)}$$

for all $x \in \{0,1\}^*$ where $\epsilon(\mathcal{P}^*, x) = \Pr\big(\mathcal{V}^{\mathsf{RO}}(x, \mathcal{P}^{*,\mathsf{RO}}(x)) = 1\big)$. Here, $\mathcal{E}$ implements RO for $\mathcal{P}$, in particular, $\mathcal{E}$ can arbitrarily program RO. Moreover, the randomness is over the randomness of $\mathcal{E}$, $\mathcal{V}$, $\mathcal{P}$ and RO.

Remark 2. Definition 9 uses a black-box extractor $\mathcal{E}$. That is, $\mathcal{E}$ has access to the next-message function of $\mathcal{P}$. Moreover, $\mathcal{E}$ does not depend on (or "know") certain properties of $\mathcal{P}^*$, such as the bound Q on the number of queries or the success probability $\epsilon(\mathcal{P}^*, x)$.

2.3 Fiat–Shamir Transformations

The Fiat-Shamir transformation [FS87] turns a public-coin interactive proof into a non-interactive random oracle proof (NIROP). The general idea is to compute the i-th challenge c_i as a hash of the i-th prover message a_i and (some part of) the previous communication transcript. For a Σ-protocol, the challenge c is computed as $c = H(a)$ or as $c = H(x, a)$, where the former is sufficient for *passive* security, where the statement x is given as input to the dishonest prover, and the latter is necessary for *adaptive* security, where the dishonest prover can choose the statement x for which it wants to forge a proof (in the latter case, Definition 9 needs to be adjusted in a non-trivial manner).

For multi-round public-coin interactive proofs, there is some degree of freedom in the computation of the i-th challenge. For concreteness and simplicity, we consider a particular version where all previous prover messages are hashed along with the current message. Also, since we consider passive security, we do not include the statement x in the hash, but including it has no negative effect on our results: all arguments go through unchanged.

Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a $(2\mu+1)$-move public-coin interactive proof, where the challenge from the i-th round is sampled from set $\mathcal{C}_i$. For simplicity, we consider μ random oracles $\mathsf{RO}_i \colon \{0,1\}^{\leq u} \to \mathcal{C}_i$ that map into the respective challenge spaces.

Definition 10. *The Fiat-Shamir transformation $\mathsf{FS}[\Pi] = (\mathcal{P}_{\mathsf{fs}}, \mathcal{V}_{\mathsf{fs}})$ is the NIROP where $\mathcal{P}_{\mathsf{fs}}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}(x;w)$ runs $\mathcal{P}(x;w)$ but instead of asking the verifier for the challenge c_i on message a_i, the challenge is computed as*

$$c_i = \mathsf{RO}_i(a_1, \ldots, a_{i-1}, a_i)\,, \tag{3}$$

where c_0 is the empty string; the output is then the proof $\pi = (a_1, \ldots, a_{\mu+1})$. Naturally, on input a statement x and a proof π, $\mathcal{V}_{\mathsf{fs}}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\mu}(x, \pi)$ computes the challenges c_i as above and outputs $\mathcal{V}(x, (a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1}))$.

By means of reducing the security of other variants of the Fiat-Shamir transformation to Definition 10, appropriately adjusted versions of our results also apply to other variants of doing the "chaining" (3) in the Fiat-Shamir transformation, for instance when c_i is computed as $c_i = \mathsf{RO}_i(i, c_{i-1}, a_i)$.

2.4 Negative Hypergeometric Distribution

Consider a bucket containing ℓ green balls and $N - \ell$ red balls, i.e., a total of N balls. In the negative hypergeometric experiment balls are drawn uniformly at random from this bucket, without replacement, until k green balls have been found or until the bucket is empty. The number of red balls X drawn in this experiment is said have a *negative hypergeometric distribution* with parameters N, ℓ, k, which is denoted by $X \sim \mathsf{NHG}(N, \ell, k)$.

Lemma 1 (Negative Hypergeometric Distribution). *Let $N, \ell, k \in \mathbb{N}$ with $\ell, k \leq N$, and let $X \sim \mathsf{NHG}(N, \ell, k)$. Then $\mathbb{E}[X] \leq k\frac{N-\ell}{\ell+1}$.*

Proof. If $\ell < k$, it clearly holds that $\Pr(X = N - \ell) = 1$. Hence, in this case, $\mathbb{E}[X] = N - \ell \leq k\frac{N-\ell}{\ell+1}$, which proves the claim.

So let us now consider the case $\ell \geq k$. Then, for all $0 \leq x \leq N - \ell$,

$$\Pr(X = x) = \frac{\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell}}\,.$$

Hence,

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{x=0}^{N-\ell} \Pr(X = x) \cdot x = \sum_{x=1}^{N-\ell} x\frac{\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell}} \\
&= k\frac{N-\ell}{\ell+1}\sum_{x=1}^{N-\ell} \frac{\frac{x}{k}\binom{x+k-1}{x}\binom{N-x-k}{N-\ell-x}}{\frac{N-\ell}{\ell+1}\binom{N}{N-\ell}} = k\frac{N-\ell}{\ell+1}\sum_{x=1}^{N-\ell} \frac{\binom{x+k-1}{x-1}\binom{N-x-k}{N-\ell-x}}{\binom{N}{N-\ell-1}} \\
&= k\frac{N-\ell}{\ell+1}\sum_{x=1}^{N-\ell} \Pr(Y = x-1) = k\frac{N-\ell}{\ell+1}\,,
\end{aligned}$$

where $Y \sim \mathsf{NHG}(N, \ell+1, k-1)$. This completes the proof of the lemma. □

Remark 3. Typically, negative hypergeometric experiments are restricted to the non-trivial case $\ell \geq k$. For reasons to become clear later, we also allow parameter choices with $\ell < k$ resulting in a trivial negative hypergeometric experiment in which all balls are always drawn.

Remark 4. The above negative hypergeometric experiment has a straightforward generalization to buckets with balls of more than 2 colors. Namely, say the bucket contain ℓ green balls and m_i balls of color i for $1 \leq i \leq M$. The experiment proceeds as before, i.e., drawing until either k green balls have been found or the bucket is empty. Let X_i be the number of balls of color i that are drawn in this experiment. Then $X_i \sim \mathsf{NHG}(\ell + m_i, \ell, k)$ for all i. To see this, simply run the generalized negative hypergeometric experiment while ignoring all balls that are neither green nor of color i.

3 An Abstract Sampling Game

Towards the goal of constructing and analyzing a knowledge extractor for the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of special-sound interactive proofs $\Pi = (\mathcal{P}, \mathcal{V})$, we define and analyze an abstract sampling game. On input a deterministic Q-query prover $\mathcal{P}^*$, attacking the non-interactive random oracle proof $\mathsf{FS}[\Pi]$, our extractor will essentially play this abstract game. The abstraction allows us to focus on the crucial properties of the extraction algorithm, without unnecessarily complicating the notation.

The game considers an arbitrary but fixed U-dimensional array M, where each entry $M(j_1, \ldots, j_U) = (v, i)$ contains a bit $v \in \{0, 1\}$ and an index $i \in \{1, \ldots, U\}$. Think of the bit v indicating whether this entry is "good" or "bad", and the index i points to one of the U dimensions. The goal will be to find k "good" entries with the same index i, and with all of them lying in the 1-dimensional array $M(j_1, \ldots, j_{i-1}, \cdot, j_{i+1}, \ldots, j_U)$ for some $j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_U$. Looking ahead, this goal is motivated by our extractor trying to find k valid proofs with different choices for the crucial challenge (and only for the crucial challenge).

To capture the main properties of the abstract sampling game, for all $1 \leq i \leq U$, we define the function

$$a_i \colon \{1, \ldots, N\}^U \to \mathbb{N}_{\geq 0}, \quad (j_1, \ldots, j_U) \mapsto \left|\left\{j : M(j_1, \ldots, j_{i-1}, j, j_{i+1}, \ldots, j_U) = (1, i)\right\}\right|\,. \tag{4}$$

The value $a_i(j_1, \ldots, j_U)$ counts the number of entries that are "good" and have index i in the 1-dimensional array $M(j_1, \ldots, j_{i-1}, \cdot, j_{i+1}, \ldots, j_U)$. Note that a_i ignores the i-th entry of the input vector $(j_1, \ldots, j_U)$. However, for notational convenience, we still consider a_i as a function acting on U-dimensional input vectors.

The game is formally defined in Figure 2 and its core properties are summarized in Lemma 2.

Fig. 2. Abstract Sampling Game.

Parameters: $k, N, U \in \mathbb{N}$ and M a U-dimensional array with entries in $M(j_1, \ldots, j_U) \in \{0,1\} \times \{1, \ldots, U\}$ for all $1 \leq j_1, \ldots, j_U \leq N$.

- Sample $(j_1, \ldots, j_U) \in \{1, \ldots, N\}^U$ uniformly at random and determine $(v, i) = M(j_1, \ldots, j_U)$.
- If $v = 0$, abort.
- Else, repeat
  - sample $j' \in \{1, \ldots, N\} \setminus \{j_i\}$ (without replacement),
  - compute $(v', i') = M(j_1, \ldots, j_{i-1}, j', j_{i+1}, \ldots, j_U)$,

  until either $k-1$ additional entries equal to $(1, i)$ have been found or until all indices j' have been tried.

Lemma 2 (Abstract Sampling Game). *Consider the game in Figure 2. Let $J = (J_1, \ldots, J_U)$ be uniformly distributed in $\{1, \ldots, N\}^U$, indicating the first entry sampled, and let $(V, I) = M(J_1, \ldots, J_U)$. Further, for all $1 \leq i \leq U$, let $A_i = a_i(J)$. Moreover, let X be the number of entries of the form $(1, i)$ with $i = I$ sampled (including the first one), and let Λ be the total number of entries sampled in this game.*

Then

$$\mathbb{E}[\Lambda] \leq 1 + (k-1)P \quad \textit{and}$$
$$\Pr(X = k) \geq \frac{N}{N-k+1}\left(\Pr(V=1) - P \cdot \frac{k-1}{N}\right),$$

where and $P = \sum_{i=1}^{U} \Pr(A_i > 0)$.

Proof. **Expected Number of Samples.** Let us first derive an upper bound on the expected value of Λ. To this end, let Y denote the number of sampled entries of the form (v, i) where $v = 0$ or $i \neq I$. Then $\Lambda = X + Y$.

Moreover, it holds that $\Pr\big(X = 0 \mid V = 0\big) = 1$ and $\Pr\big(X \leq k \mid V = 1\big) = 1$. Hence,

$$\begin{aligned}\mathbb{E}[X] &= \Pr(V=0) \cdot \mathbb{E}[X \mid V = 0] + \Pr(V=1) \cdot \mathbb{E}[X \mid V = 1] \\ &\leq \Pr(V=1) \cdot k\,.\end{aligned}$$

So let us now consider the random variable Y. Then, conditioned on "$V = 1 \wedge I = i \wedge A_i = a$", Y follows a negative hypergeometric distribution with parameters $N-1$, $a-1$ and $k-1$. Hence, by Lemma 1,

$$\mathbb{E}[Y \mid V = 1 \wedge I = i \wedge A_i = a] \leq (k-1)\frac{N-a}{a}\,.$$

Hence,

$$\begin{aligned}\mathbb{E}[Y] &= \Pr(V=0) \cdot \mathbb{E}[Y \mid V = 0] + \Pr(V=1) \cdot \mathbb{E}[Y \mid V = 1] \\ &= \Pr(V=0) + \sum_{i=1}^{U} \Pr(V = 1 \wedge I = i)\, \mathbb{E}[Y \mid V = 1 \wedge I = i] \\ &= \Pr(V=0) + \sum_{i=1}^{U}\sum_{a=0}^{N} \Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)\, \mathbb{E}[Y \mid V = 1 \wedge I = i \wedge A_i = a] \\ &\leq \Pr(V=0) + (k-1)\sum_{i=1}^{U}\sum_{a=1}^{N} \Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)\frac{N-a}{a}\,,\end{aligned} \tag{5}$$

where we additionally use that $\mathbb{E}[Y \mid V = 0] = 1$ and $\Pr\big(V = 1 \wedge I = i \wedge A_i = 0\big) = 0$ for all $1 \leq i \leq U$.

Now note that, for any $1 \leq i \leq U$ and $0 \leq a \leq N$,

$$\Pr\big(V = 1 \wedge I = i \mid A_i = a\big) = \frac{a}{N}\,, \tag{6}$$

Hence,

$$\begin{aligned}\Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)\frac{N-a}{a} &= \frac{\Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)}{\Pr\big(V = 1 \wedge I = i \mid A_i = a\big)} - \Pr\big(V = 1 \wedge I = i \wedge A_i = a\big) \\ &= \Pr(A_i = a) - \Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)\,.\end{aligned}$$

Therefore, combined with Equation 5, it follows that

$$\begin{aligned}\mathbb{E}[Y] &\leq \Pr(V = 0) + (k-1)\sum_{i=1}^{U}\sum_{a=1}^{N}\Big(\Pr(A_i = a) - \Pr\big(V = 1 \wedge I = i \wedge A_i = a\big)\Big) \\ &= \Pr(V = 0) + (k-1)\big(P - \Pr(V = 1)\big) \\ &= \Pr(V = 0) + (k-1)\big(P - 1 + \Pr(V = 0)\big) \\ &= k\Pr(V = 0) + (k-1)(P-1)\,,\end{aligned}$$

where $P = \sum_{i=1}^{U}\Pr(A_i > 0)$.

Hence,

$$\begin{aligned}\mathbb{E}[\Lambda] = \mathbb{E}[X] + \mathbb{E}[Y] &\leq k \cdot \Pr(V = 1) + k \cdot \Pr(V = 0) + (k-1)(P-1) \\ &= k + (k-1)(P-1) = 1 + P(k-1)\,,\end{aligned}$$

which proves the claimed upper bound on $\mathbb{E}[\Lambda]$.

Success Probability. Let us now find a lower bound for the "success probability" $\Pr(X = k)$ of this game. By basic probability theory, it follows that

$$\begin{aligned}\Pr(X = k) &= \sum_{i=1}^{U}\Pr(V = 1 \wedge I = i \wedge A_i \geq k) \\ &= \Pr(V = 1) - \sum_{i=1}^{U}\Pr(V = 1 \wedge I = i \wedge 0 < A_i < k) \\ &= \Pr(V = 1) - \sum_{i=1}^{U}\Pr(0 < A_i < k) \cdot \Pr(V = 1 \wedge I = i \mid 0 < A_i < k) \\ &\geq \Pr(V = 1) - \frac{k-1}{N}\sum_{i=1}^{U}\Pr(0 < A_i < k)\,,\end{aligned}$$

where we have used that $\Pr(V = 1 \wedge I = i \wedge A_i = 0) = 0$ and $\Pr(V = 1 \wedge I = i \mid 0 < A_i < k) \leq (k-1)/N$.[4]

Now note that, for any i,

$$\begin{aligned}(N-k+1)\Pr(0 < A_i < k) &= N\left(\Pr(A_i > 0) - \Pr(A_i \geq k)\right) - (k-1)\Pr(0 < A_i < k) \\ &= N\Pr(A_i > 0) - N\Pr(A_i \geq k) - (k-1)\Pr(0 < A_i < k) \\ &\leq N\Pr(A_i > 0) - N\Pr(V = 1 \wedge I = i \mid A_i \geq k)\Pr(A_i \geq k) - \\ &\qquad N\Pr(V = 1 \wedge I = i \mid 0 < A_i < k)\Pr(0 < A_i < k) \\ &= N\Pr(A_i > 0) - N\Pr(V = 1 \wedge I = i)\,,\end{aligned}$$

[4] At this stage of the proof, it already follows that the success probability satisfies $\Pr(X = k) \geq \Pr(V = 1) - P\frac{k-1}{N}$. This lower bound is sufficient to prove knowledge soundness of the Fiat-Shamir transformation of 3-move k-special-sound protocols. To (tightly) handle multi-round protocols, the larger lower-bound of our lemma is required. For this reason, we continue to prove the stronger result as stated in the lemma.

where, for the inequality, we use that $\Pr(V = 1 \wedge I = i \mid A_i \geq k) \leq 1$ and, once more, $\Pr(V = 1 \wedge I = i \mid 0 < A_i < k) \leq (k-1)/N$. Further, for the final inequality we use that $\Pr(V = 1 \wedge I = i \mid A_i = 0) = 0$.

Hence, combining the above equations, it follows that

$$\begin{aligned}\Pr(X = k) &\geq \Pr(V = 1) - \frac{k-1}{N-k+1} \sum_{i=1}^{U} \Big(\Pr(A_i > 0) - \Pr(V = 1 \wedge I = i) \Big) \\ &= \Pr(V = 1) - \frac{k-1}{N-k+1} \left(P - \Pr(V = 1) \right) \\ &= \frac{N}{N-k+1} \left(\Pr(V = 1) - P \cdot \frac{k-1}{N} \right) ,\end{aligned}$$

where, as before, $P = \sum_{i=1}^{U} \Pr(A_i > 0)$. This completes the proof of the lemma.

□

Our knowledge extractor will instantiate the abstract sampling game via a deterministic Q-query prover $\mathcal{P}^*$ attacking the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of an interactive proof Π. In this instantiation, the quantity $P = \sum_{i=1}^{U} \Pr(A_i > 0)$ of Lemma 2 corresponds to the security loss of the Fiat-Shamir transformation. In general, P has U as a *tight* upper bound, which is insufficient for our purposes. However, the following lemma shows that for certain instantiations of the abstract sampling game P has a much smaller upper bound. It turns out that these are precisely the instantiations of our knowledge extractor.

Lemma 3. *Consider the game in Figure 2. Let V, I be functions such that $M(\mathbf{j}) = \big(V(\mathbf{j}), I(\mathbf{j})\big)$ for all $\mathbf{j} \in \{1, \ldots, N\}^U$, and let $J = (J_1, \ldots, J_U)$ be uniformly distributed in $\{1, \ldots, N\}^U$. Further, for all $1 \leq i \leq U$, let $A_i = a_i(J)$.*

Let us additionally assume that for all $\mathbf{j} \in \{1, \ldots, N\}^U$ there exists a subset $S(\mathbf{j}) \subset \{1, \ldots, U\}$ of cardinality at most Q such that $I(\mathbf{j}) = I(\mathbf{j}')$ for all $\mathbf{j}, \mathbf{j}'$ with $j_i = j'_i$ for all $i \in S(\mathbf{j})$. Then

$$P = \sum_{i=1}^{U} \Pr(A_i > 0) \leq Q + 1 \, .$$

Proof. By basic probability theory, it follows that

$$\begin{aligned}P &= \sum_{i=1}^{U} \Pr(A_i > 0) \\ &= \sum_{\mathbf{j} \in \{1,\ldots,N\}^U} \Pr(J = \mathbf{j}) \sum_{i=1}^{U} \Pr(A_i > 0 \mid J = \mathbf{j}) \\ &= \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \left(\sum_{i \in S(\mathbf{j})} \Pr(A_i > 0 \mid J = \mathbf{j}) + \sum_{i \notin S(\mathbf{j})} \Pr(A_i > 0 \mid J = \mathbf{j}) \right) \\ &\leq \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \left(Q + \sum_{i \notin S(\mathbf{j})} \Pr(A_i > 0 \mid J = \mathbf{j}) \right) \\ &\leq Q + \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \sum_{i \notin S(\mathbf{j})} \Pr(A_i > 0 \mid J = \mathbf{j}) \, ,\end{aligned}$$

where the inequality follows from the fact that $|S(\mathbf{j})| \leq Q$ for all $\mathbf{j}$.

Now note that, by definition of the sets $S(\mathbf{j})$, for all $\mathbf{j} \in \{1, \ldots, N\}^U$, $j \in \{1, \ldots, N\}$ and $i \notin S(\mathbf{j})$, it holds that

$$\Pr\big(I(J_1, \ldots, J_{i-1}, j, J_{i+1}, \ldots, J_U) = I(\mathbf{j}) \mid J = \mathbf{j}\big) = 1 \, .$$

Therefore, for all $i \notin S(\mathbf{j}) \cup \{I(\mathbf{j})\}$,

$$\Pr(A_i > 0 \mid J = \mathbf{j}) = 0\,.$$

Hence,

$$\sum_{i \notin S(\mathbf{j})} \Pr(A_i > 0 \mid J = \mathbf{j}) \leq \Pr(A_{I(\mathbf{j})} > 0 \mid J = \mathbf{j}) \leq 1.$$

Altogether, it follows that

$$P \leq Q + \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) = Q + 1\,,$$

which completes the proof. □

4 Fiat-Shamir Transformation of Σ-Protocols

Let us first consider the Fiat-Shamir transformation of a k-special-sound Σ-protocol Π, i.e., a 3-move interactive proof, with challenge set $\mathcal{C}$; subsequently, in Section 6, we move to general *multi-round* interactive proofs.

Let $\mathcal{P}^*$ be a deterministic dishonest Q-query random-oracle prover, attacking the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of Π on input x. Given a statement x as input, after making Q queries to the random oracle $\mathsf{RO}\colon \{0,1\}^{\leq u} \to \mathcal{C}$, $\mathcal{P}^*$ outputs a proof $\pi = (a, z)$. For reasons to become clear later, we re-format (and partly rename) the output and consider

$$I := a \qquad \text{and} \qquad \pi$$

as $\mathcal{P}^*$'s output. We refer to the output I as the *index*.

Furthermore, we extend $\mathcal{P}^*(x)$ to an algorithm $\mathcal{A}$ that additionally checks the correctness of the proof π. Formally, $\mathcal{A}$ runs $\mathcal{P}^*(x)$ to obtain I and π, queries RO to obtain $c := \mathsf{RO}(I)$, and then outputs

$$I = a\,, \quad y := (a, c, z) \qquad \text{and} \qquad v := V(y)\,,$$

where $V(y) = 1$ if y is an accepting transcript for the interactive proof Π on input x and $V(y) = 0$ otherwise. We will also write $\mathcal{A}^{\mathsf{RO}}$ for the algorithm that executes $\mathcal{A}$ given a *fixed* random oracle RO.

Hence, $\mathcal{A}$ is a random-oracle algorithm making at most $Q + 1$ queries; indeed, it relays the oracle queries done by $\mathcal{P}^*$ and, if necessary, makes the one needed to do the verification. Moreover, $\mathcal{A}$ has a naturally defined success probability

$$\epsilon(\mathcal{A}) := \Pr\bigl(v = 1 : (I, y, v) \leftarrow \mathcal{A}^{\mathsf{RO}}\bigr)\,,$$

where $\mathsf{RO}\colon \{0,1\}^{\leq u} \to \mathcal{C}$ is chosen uniformly at random. The success probability $\epsilon(\mathcal{A})$ corresponds to the success probability $\epsilon(\mathcal{P}^*, x)$ of the random-oracle prover $\mathcal{P}^*$ on input x.

Our goal is now to construct an extraction algorithm that, when given black-box access to $\mathcal{A}$, aims to output k accepting transcripts $y_1, \ldots, y_k$ with common first message a and distinct challenges. By the k-special-soundness property of Π, a witness for statement x can be computed efficiently from these transcripts.

The extractor $\mathcal{E}$ is defined in Figure 3. We remark that, by construction of $\mathcal{A}$, $\mathcal{A}$ does make a query to I; thus, c_i is well defined in Figure 3. Also, since $\mathcal{P}^*$ and thus $\mathcal{A}$ is deterministic, in each iteration of the repeat loop $\mathcal{A}$ is ensured to make the query to I again.

A crucial observation is the following. Within a run of $\mathcal{E}$, all the queries that are made by the different invocations of $\mathcal{A}$ are answered *consistently* using lazy sampling, except for the queries to the index i, where different responses $c_i, c_i', \ldots$ are given. This is indistinguishable from having them answered by a full-fledged random oracle, i.e., by means of a pre-chosen function $\mathsf{RO} : \{0,1\}^{\leq u} \to \mathcal{C}$, $\ell \mapsto j_\ell = \mathsf{RO}(\ell)$, but then replacing j_i by fresh $j' \neq j_i$ for the runs of $\mathcal{A}$ in the repeat loop. Thus, the extractor is actually running the abstract game from Figure 2 and bounds on the success probability and the expected run time (in terms of queries to $\mathcal{A}$) follow from Lemma 2 and Lemma 3. Altogether we obtain the following result.

Fig. 3. Extractor $\mathcal{E}$.

Parameters: $k, Q \in \mathbb{N}$
Black-box access to: $\mathcal{A}$ as above

- Run $\mathcal{A}$ as follows to obtain (I, y_1, v): answer all (distinct) oracle queries with uniformly random values in $\mathcal{C}$. Set $i := I$, and let c_i be the response to query i.
- If $v = 0$, abort.
- Else, repeat
  - sample $c_i' \in \mathcal{C} \setminus \{c_i\}$ (without replacement);
  - run $\mathcal{A}$ as follows to obtain (I', y', v'): answer the query to i with c_i', while answering all other queries consistently if the query was performed by $\mathcal{A}$ already on a previous run and with a fresh random value in $\mathcal{C}$ otherwise;

  until either $k-1$ additional challenges c_i' with $v' = 1$ and $I' = I$ have been found or until all challenges $c_i' \in \mathcal{C}$ have been tried.
- In the former case, output the k accepting transcripts $y_1, \ldots, y_k$.

Lemma 4 (Extractor). *The extractor $\mathcal{E}$ of Figure 3 makes an expected number of at most $k + Q(k-1)$ queries to $\mathcal{A}$ and succeeds in outputting k transcripts $y_1, \ldots, y_k$ with common first message a and distinct challenges with probability at least*

$$\frac{N}{N-k+1}\left(\epsilon(\mathcal{A}) - (Q+1)\cdot\frac{k-1}{N}\right).$$

Proof. Let U be the cardinality of the domain $\{0,1\}^{\leq u}$ of a random oracle $\mathsf{RO}\colon \{0,1\}^{\leq u} \to \mathcal{C}$. Considering an arbitrary but fixed ordering $\xi_1, \ldots, \xi_U$ of the bitstrings $\xi_i \in \{0,1\}^{\leq u}$, a vector $\mathbf{c} \in \mathcal{C}^U$ then encodes the function table of the entire random oracle as $\mathsf{RO}(\xi_i) = c_i$. For this reason, we can also refer to $\mathbf{c} \in \mathcal{C}^U$ as a (full-fledged) random oracle.

Further, since $\mathcal{P}^*$ is deterministic, the outputs I, y and v of the algorithm $\mathcal{A}$ can be viewed as functions taking as input a random oracle $\mathbf{c} \in \mathcal{C}^U$.

Let us now consider the following array $M(\mathbf{c}) = \big(I(\mathbf{c}), v(\mathbf{c})\big)$, indexed by random oracles $\mathbf{c} \in \mathcal{C}^U$. Then, a single run of the extractor is indistinguishable from playing the abstract sampling game of Figure 2 instantiated with array M. The only difference is that, in this sampling game, we consider full-fledged random oracles encoded by vectors $\mathbf{c} \in \mathcal{C}^U$, while the actual extractor implements these random oracles by lazy sampling.

Thus, we can apply Lemma 2 to obtain bounds on the success probability and the expected run time. However, in order to control the parameter P, which occurs in the bound of Lemma 2, we make the following observation, so that we can apply Lemma 3 and prove that $P \leq Q + 1$.

Namely, since $\mathcal{P}^*$ is deterministic, its output can only change when the random oracle is reprogrammed at one of the indices $i \in \{0,1\}^{\leq u}$ queried by $\mathcal{P}^*$. In other words, for every random oracle $\mathbf{c} \in \mathcal{C}^U$, there exists a subset $S(\mathbf{c}) \subset \{0,1\}^{\leq u}$ (indicating the queries made by $\mathcal{P}^*$) such that $\mathcal{P}^*$'s output stays the same when the random oracle is reprogrammed at an index $i \notin S(\mathbf{c})$. In particular, $I(\mathbf{c}) = I(\mathbf{c}')$ for all $\mathbf{c}, \mathbf{c}'$ with $c_i = c_i'$ for all $i \in S(\mathbf{c})$. Hence, the conditions of Lemma 3 are satisfied and $P \leq Q + 1$. The bounds on the success probability and the expected run time now follow, which completes the proof. □

Given the existence of the above extractor, combined with the k-special-soundness property, implies the following theorem.

Theorem 1 (Fiat-Shamir Transformation of a Σ-Protocol). *The Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of a k-out-of-N special-sound Σ-protocol Π is knowledge sound with knowledge error*

$$\kappa_{\mathrm{fs}}(Q) = (Q+1)\cdot\kappa\,,$$

where $\kappa := \mathrm{Er}(k; N) = (k-1)/N$ is the knowledge error of the (interactive) Σ-protocol Π.

5 Refined Analysis of the Abstract Sampling Game

Before we prove knowledge soundness of the Fiat-Shamir transformation of *multi-round* interactive protocols, we reconsider the abstract game of Section 3, and consider a refined analysis of the costs of playing the game. The multi-round knowledge extractor will essentially play a recursive composition of this game; however, the analysis of Section 3 is insufficient for our purposes (resulting in a super-polynomial bound on the run-time of the knowledge extractor). Fortunately, it turns out that a refinement allows us to prove the required (polynomial) upper bounds.

In Section 3, the considered cost measure is the number of entries visited during the game. For Σ-protocols, every entry corresponds to a single invocation of the dishonest prover $\mathcal{P}^*(x)$. For multi-round protocols, every entry will correspond to a single invocation of a *subtree extractor*. The key observation is that some invocations of the subtree extractor are *expensive* while others are *cheap*. For this reason, we introduce a cost function Γ and a constant cost γ to our abstract game, allowing us to differentiate between these two cases. The functions assign a cost to every entry of the array M; Γ corresponds to the cost of an expensive invocation of the subtree extractor and γ corresponds to the costs of a cheap invocation.

The following lemma provides an upper bound for the total costs of playing the abstract game in terms of these two cost functions.

Lemma 5 (Abstract Sampling Game - Weighted Version). *Consider again the game of Figure 2, as well a cost function $\Gamma\colon \{1,\dots,N\}^U \to \mathbb{R}_{\geq 0}$ and a constant cost $\gamma \in \mathbb{R}_{\geq 0}$. Let $J = (J_1,\dots,J_U)$ be uniformly distributed in $\{1,\dots,N\}^U$, indicating the first entry sampled, and let $(V,I) = M(J_1,\dots,J_U)$. Further, for all $1 \leq i \leq U$, let $A_i = a_i(J)$, where the function a_i is as defined in Equation 4.*

We define the cost of sampling an entry $M(j_1,\dots,j_U) = (v,i)$ with index $i = I$ to be $\Gamma(j_1,\dots,j_U)$ and the cost of sampling an entry $M(j_1,\dots,j_U) = (v,i)$ with index $i \neq I$ to be γ. Let Δ be the total cost of playing this game. Then

$$\mathbb{E}[\Delta] \leq k \cdot \mathbb{E}[\Gamma(J)] + (k-1) \cdot T \cdot \gamma\,.$$

where $T = \sum_{i=1}^{U} \Pr(I \neq i \wedge A_i > 0)$.

Proof. Let us define three more fine-grained cost measures Δ_1, Δ_2 and Δ_3. First, Δ_1 denotes the total costs of the elements $M(j_1,\dots,j_U) = (1,i)$ with $i = I$ sampled in the game, i.e., the elements with bit $v = 1$ and index $i = I$. Correspondingly, X denotes the number of entries of the form $(1,i)$ with $i = I$ sampled. Second, Δ_2 denotes the total costs of the elements $M(j_1,\dots,j_U) = (0,i)$ with $i = I$ sampled in the game, i.e., the elements with bit $v = 0$ and index $i = I$. Correspondingly, Y denotes the number of entries of the form $(0,i)$ with $i = I$ sampled. Finally, Δ_3 denotes the total costs of the elements $M(j_1,\dots,j_U) = (v,i)$ with $i \neq I$ sampled in this game. Correspondingly, Z denotes the number of entries of this form sampled. Then, clearly $\Delta = \Delta_1 + \Delta_2 + \Delta_3$.

For all $1 \leq i \leq U$, let us write $J_i^* = (J_1,\dots,J_{i-1},J_{i+1},\dots,J_U)$ for the uniformly distributed random variable with support $\{1,\dots,N\}^{U-1}$. Moreover, for all $1 \leq i \leq U$ and $j^* \in \{1,\dots,N\}^{U-1}$, let $\Lambda(i,j^*)$ denote the event

$$I = i \wedge J_i^* = j^* \wedge A_i > 0\,.$$

Then

$$\begin{aligned}\mathbb{E}[\Delta_1 \mid \Lambda(i,j^*)] &= \sum_{\ell=0}^{N} \Pr\big(X = \ell \mid \Lambda(i,j^*)\big) \cdot \mathbb{E}[\Delta_1 \mid \Lambda(i,j^*) \wedge X = \ell] \\ &= \sum_{\ell=0}^{N} \Pr\big(X = \ell \mid \Lambda(i,j^*)\big) \cdot \ell \cdot \mathbb{E}[\Delta_1/\ell \mid \Lambda(i,j^*) \wedge X = \ell]\end{aligned}$$

Since, conditioned on "$\Lambda(i,j^*) \wedge X = \ell$", any subset of elements of the Δ_1-type of cardinality ℓ is equally likely to be sampled, it follows that

$$\mathbb{E}[\Delta_1/\ell \mid \Lambda(i,j^*) \wedge X = \ell] = \mathbb{E}[\Gamma(J) \mid V = 1 \wedge \Lambda(i,j^*)]\,.$$

Hence,

$$\begin{aligned}\mathbb{E}[\Delta_1 \mid \Lambda(i,j^*)] &= \mathbb{E}[\Gamma(J) \mid V = 1 \wedge \Lambda(i,j^*)] \cdot \sum_{\ell=0}^{N} \Pr\big(X = \ell \mid \Lambda(i,j^*)\big) \cdot \ell \\ &= \mathbb{E}[\Gamma(J) \mid V = 1 \wedge \Lambda(i,j^*)] \cdot \mathbb{E}[X \mid \Lambda(i,j^*)]\,.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}[\Delta_2 \mid \Lambda(i,j^*)] &= \mathbb{E}[\Gamma(J) \mid V = 0 \wedge \Lambda(i,j^*)] \cdot \mathbb{E}[Y \mid \Lambda(i,j^*)]\,, \\ \mathbb{E}[\Delta_3 \mid \Lambda(i,j^*)] &= \gamma \cdot \mathbb{E}[Z \mid \Lambda(i,j^*)]\,,\end{aligned}$$

using that γ is constant.

Moreover, since $\Pr(V = 0 \mid I = i \wedge A_i = 0) = 1$, it follows that

$$\mathbb{E}[\Delta \mid I = i \wedge A_i = 0] = \mathbb{E}[\Gamma(J) \mid I = i \wedge A_i = 0]\,.$$

Hence,

$$\begin{aligned}\mathbb{E}[\Delta] &= \sum_{i=1}^{U} \Big(\Pr(I = i \wedge A_i = 0) \cdot \mathbb{E}[\Delta \mid I = i \wedge A_i = 0] + \Pr(I = i \wedge A_i > 0) \cdot \mathbb{E}[\Delta \mid I = i \wedge A_i > 0] \Big) \\ &= \sum_{i=1}^{U} \Big(\Pr(I = i \wedge A_i = 0) \cdot \mathbb{E}[\Gamma(J) \mid I = i \wedge A_i = 0] + \sum_{j^*} \Pr(\Lambda(i,j^*)) \cdot \mathbb{E}[\Delta \mid \Lambda(i,j^*)] \Big)\,.\end{aligned}$$

where the final summation is over all $j^* \in \{1,\dots,N\}^{U-1}$.

Moreover, by the above,

$$\begin{aligned}\mathbb{E}[\Delta \mid \Lambda(i,j^*)] &= \mathbb{E}[\Delta_1 \mid \Lambda(i,j^*)] + \mathbb{E}[\Delta_2 \mid \Lambda(i,j^*)] + \mathbb{E}[\Delta_3 \mid \Lambda(i,j^*)] \\ &= \mathbb{E}[\Gamma(J) \mid V = 1 \wedge \Lambda(i,j^*)] \cdot \mathbb{E}[X \mid \Lambda(i,j^*)] \\ &\quad + \mathbb{E}[\Gamma(J) \mid V = 0 \wedge \Lambda(i,j^*)] \cdot \mathbb{E}[Y \mid \Lambda(i,j^*)] \\ &\quad + \gamma \cdot \mathbb{E}[Z \mid \Lambda(i,j^*)]\,.\end{aligned} \tag{7}$$

For this reason, let us consider the expected values of X, Y and Z conditioned on $\Lambda(i,j^*)$. The analysis is a more fine-grained version of the proof of Lemma 2.

Since $\Pr\big(X = 0 \mid V = 0 \wedge \Lambda(i,j^*)\big) = 1$ and $\Pr\big(X \le k \mid V = 1 \wedge \Lambda(i,j^*)\big) = 1$, it immediately follows that

$$\begin{aligned}\mathbb{E}[X \mid \Lambda(i,j^*)] &= \Pr(V = 0 \mid \Lambda(i,j^*)) \cdot \mathbb{E}[X \mid V = 0 \wedge \Lambda(i,j^*)] \\ &\quad + \Pr(V = 1 \mid \Lambda(i,j^*)) \cdot \mathbb{E}[X \mid V = 1 \wedge \Lambda(i,j^*)] \\ &\le \Pr(V = 1 \mid \Lambda(i,j^*)) \cdot k\,.\end{aligned}$$

So let us now consider the random variables Y and Z, again conditioned on $\Lambda(i,j^*)$. For given $1 \le i \le U$ and $j^* = (j_1^*,\dots,j_{U-1}^*)$, we let

$$a = \big|\big\{ j \,\big|\, V_j = 1 \wedge I_j = i : (V_j, I_j) = M(j_1^*,\dots,j_{i-1}^*, j, j_i^*,\dots,j_{U-1}^*) \big\}\big|\,,$$

and

$$b = \big|\big\{ j \,\big|\, V_j = 0 \wedge I_j = i : (V_j, I_j) = M(j_1^*,\dots,j_{i-1}^*, j, j_i^*,\dots,j_{U-1}^*) \big\}\big|\,.$$

Note that, a equals the random variable A_i conditioned on $J_i^* = j^*$ or, more precisely, $\Pr(A_i = a \mid J_i^* = j^*) = 1$.

Then, conditioned on "$V = 1 \wedge \Lambda(i,j^*)$", Y follows a negative hypergeometric distribution with parameters $a + b - 1$, $a - 1$ and $k - 1$ (see also Remark 4). Hence, by Lemma 1,

$$\mathbb{E}[Y \mid V = 1 \wedge \Lambda(i,j^*)] \le (k-1)\frac{b}{a}\,.$$

Note that, conditioned on $A_i > 0$, it holds that $a > 0$, which shows that the above inequality is well-defined.

Similarly, conditioned on "$V = 1 \wedge \Lambda(i, j^*)$", Z follows a negative hypergeometric distribution with parameters $N - b - 1$, $a - 1$ and $k - 1$ and

$$\mathbb{E}[Z \mid V = 1 \wedge \Lambda(i, j^*)] \leq (k-1)\frac{N-a-b}{a}\,.$$

Hence,

$$\begin{aligned}\mathbb{E}[Y \mid \Lambda(i, j^*)] &= \Pr(V = 0 \mid \Lambda(i, j^*)) \cdot \mathbb{E}[Y \mid V = 0 \wedge \Lambda(i, j^*)] \\ &\quad + \Pr(V = 1 \mid \Lambda(i, j^*)) \cdot \mathbb{E}[Y \mid V = 1 \wedge \Lambda(i, j^*)] \\ &\leq \Pr\big(V = 0 \mid \Lambda(i, j^*)\big) + \Pr\big(V = 1 \mid \Lambda(i, j^*)\big) \cdot (k-1)\frac{b}{a}\,,\end{aligned} \tag{8}$$

where we use that $\mathbb{E}[Y \mid V = 0 \wedge \Lambda(i, j^*)] = 1$.

Similarly, it follows that

$$\mathbb{E}[Z \mid \Lambda(i, j^*)] \leq \Pr\big(V = 1 \mid \Lambda(i, j^*)\big) \cdot (k-1)\frac{N-a-b}{a}\,, \tag{9}$$

where we now use that $\mathbb{E}[Z \mid V = 0 \wedge \Lambda(i, j^*)] = 0$.

Now note that

$$\begin{aligned}&\Pr\big(V = 1 \mid \Lambda(i, j^*)\big) = \frac{a}{a+b}\,, \\ &\Pr\big(V = 0 \mid \Lambda(i, j^*)\big) = \frac{b}{a+b} \quad \text{and} \\ &\Pr\big(V = 1 \wedge I = i \mid J_i^* = j^* \wedge A_i > 0\big) = \frac{a}{N}\,.\end{aligned} \tag{10}$$

Hence,

$$\begin{aligned}\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)\frac{b}{a} &= \Pr\big(V = 1 \mid \Lambda(i, j^*)\big)\frac{\Pr\big(V = 0 \mid \Lambda(i, j^*)\big)}{\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)} \\ &= \Pr\big(V = 0 \mid \Lambda(i, j^*)\big)\,.\end{aligned}$$

Therefore, combined with Equation 8, it follows that

$$\begin{aligned}\mathbb{E}[Y \mid \Lambda(i, j^*)] &\leq \Pr(V = 0 \mid \Lambda(i, j^*)) + (k-1) \cdot \Pr\big(V = 0 \mid \Lambda(i, j^*)\big) \\ &= k \cdot \Pr(V = 0 \mid \Lambda(i, j^*))\,.\end{aligned}$$

Moreover,

$$\begin{aligned}&\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)\frac{N-a-b}{a} \\ &\quad= \frac{\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)}{\Pr\big(V = 1 \wedge I = i \mid J_i^* = j^* \wedge A_i > 0\big)} - \frac{\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)}{\Pr\big(V = 1 \mid \Lambda(i, j^*)\big)} \\ &\quad= \frac{1}{\Pr(I = i \mid J_i^* = j^* \wedge A_i > 0)} - 1\,.\end{aligned}$$

Therefore, combined with Equation 9, it follows that

$$\mathbb{E}[Z \mid \Lambda(i, j^*)] \leq \frac{k-1}{\Pr\big(I = i \mid J_i^* = j^* \wedge A_i > 0\big)} - k + 1\,.$$

Combining the upper bounds on the expected values $\mathbb{E}[X \mid \Lambda(i,j^*)]$, $\mathbb{E}[Y \mid \Lambda(i,j^*)]$ and $\mathbb{E}[Z \mid \Lambda(i,j^*)]$ with Equation 7 shows that

$$\begin{aligned}
&\sum_{j^*} \Pr\big(\Lambda(i,j^*)\big) \cdot \mathbb{E}[\Delta \mid \Lambda(i,j^*)] \\
&\quad \leq \sum_{j^*} \Pr\big(\Lambda(i,j^*)\big) \Big(\mathbb{E}[\Gamma(J) \mid V = 1 \wedge \Lambda(i,j^*)] \cdot k \cdot \Pr\big(V = 1 \mid \Lambda(i,j^*)\big) \\
&\qquad + \mathbb{E}[\Gamma(J) \mid V = 0 \wedge \Lambda(i,j^*)] \cdot k \cdot \Pr\big(V = 0 \mid \Lambda(i,j^*)\big) \\
&\qquad + \gamma \cdot \Big(\frac{k-1}{\Pr\big(I = i \mid J_i^* = j^* \wedge A_i > 0\big)} - k + 1 \Big) \Big) \\
&\quad = k \cdot \Pr(I = i \wedge A_i > 0) \cdot \mathbb{E}[\Gamma(J) \mid I = i \wedge A_i > 0] + (k-1) \cdot \gamma \cdot \\
&\qquad \sum_{j^*} \big(\Pr(J_i^* = j^* \wedge A_i > 0) - \Pr\big(\Lambda(i,j^*)\big) \big) \,.
\end{aligned}$$

Moreover, considering only the final summation,

$$\begin{aligned}
&\sum_{i=1}^{U} \sum_{j^*} \big(\Pr(J_i^* = j^* \wedge A_i > 0) - \Pr\big(\Lambda(i,j^*)\big) \big) \\
&\qquad = \sum_{i=1}^{U} \big(\Pr(A_i > 0) - \Pr\big(I = i \wedge A_i > 0\big) \big) \\
&\qquad = \sum_{i=1}^{U} \Pr(I \neq i \wedge A_i > 0) \\
&\qquad := T \,.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}[\Delta] &\leq \sum_{i=1}^{U} \Big(\Pr(I = i \wedge A_i = 0) \cdot \mathbb{E}[\Gamma(J) \mid I = i \wedge A_i = 0] \\
&\qquad + k \cdot \Pr(I = i \wedge A_i > 0) \cdot \mathbb{E}[\Gamma(J) \mid I = i \wedge A_i > 0] \Big) \\
&\qquad + (k-1) \cdot T \cdot \gamma \\
&\leq k \cdot \mathbb{E}[\Gamma(J)] + (k-1) \cdot T \cdot \gamma \,.
\end{aligned}$$

which completes the proof.

□

Note that the factor $T = \sum_{i=1}^{U} \Pr(I \neq i \wedge A_i > 0)$ of Lemma 5 differs from its analogue $P = \sum_{i=1}^{U} \Pr(A_i > 0)$ in Lemma 2. However, the following lemma shows that T has a similar upper bound for certain instantiations of the abstract sampling game.

Lemma 6. *Consider the game in Figure 2. Let V, I be functions such that $M(\mathbf{j}) = \big(V(\mathbf{j}), I(\mathbf{j})\big)$ for all $\mathbf{j} \in \{1, \ldots, N\}^U$, and let $J = (J_1, \ldots, J_U)$ be uniformly distributed in $\{1, \ldots, N\}^U$. Further, for all $1 \leq i \leq U$, let $A_i = a_i(J)$.*

Let us additionally assume that for all $\mathbf{j} \in \{1, \ldots, N\}^U$ there exists a subset $S(\mathbf{j}) \subset \{1, \ldots, U\}$ of cardinality at most Q such that $I(\mathbf{j}) = I(\mathbf{j}')$ for all $\mathbf{j}, \mathbf{j}'$ with $j_i = j_i'$ for all $i \in S(\mathbf{j})$. Then

$$T = \sum_{i=1}^{U} \Pr(I(J) \neq i \wedge A_i > 0) \leq Q \,.$$

Proof. The proof is analogous to the proof of Lemma 3.

By basic probability theory, it follows that

$$\begin{aligned} T &= \sum_{i=1}^{U} \Pr(I(J) \neq i \wedge A_i > 0) \\ &= \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \left(\sum_{i \in S(\mathbf{j})} \Pr(I(J) \neq i \wedge A_i > 0 \mid J = \mathbf{j}) + \sum_{i \notin S(\mathbf{j})} \Pr(I(J) \neq i \wedge A_i > 0 \mid J = \mathbf{j}) \right) \\ &\leq Q + \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \sum_{i \notin S(\mathbf{j})} \Pr(I(J) \neq i \wedge A_i > 0 \mid J = \mathbf{j}) \,, \end{aligned}$$

where the inequality follows from the fact that $|S(\mathbf{j})| \leq Q$ for all $\mathbf{j}$.

Now note that, by definition of the sets $S(\mathbf{j})$, for all $\mathbf{j} \in \{1, \ldots, N\}^U$, $j \in \{1, \ldots, N\}$ and $i \notin S(\mathbf{j})$, it holds that

$$\Pr\big(I(J_1, \ldots, J_{i-1}, j, J_{i+1}, \ldots, J_U) = I(\mathbf{j}) \mid J = \mathbf{j}\big) = 1 \,.$$

Therefore, for all $i \notin S(\mathbf{j}) \cup \{I(\mathbf{j})\}$,

$$\Pr(A_i > 0 \mid J = \mathbf{j}) = 0 \,.$$

Hence,

$$\sum_{i \notin S(\mathbf{j})} \Pr(I(J) \neq i \wedge A_i > 0 \mid J = \mathbf{j}) \leq \Pr(I(J) \neq I(\mathbf{j}) \wedge A_{I(\mathbf{j})} > 0 \mid J = \mathbf{j}) = 0.$$

Altogether, it follows that

$$T \leq Q + \sum_{\mathbf{j}} \Pr(J = \mathbf{j}) \sum_{i \notin S(\mathbf{j})} \Pr(I(J) \neq i \wedge A_i > 0 \mid J = \mathbf{j}) = Q \,,$$

which completes the proof. □

6 The Fiat-Shamir Transformation of Multi-Round Protocols

Let us now move to multi-round interactive proofs. More precisely, we consider the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of a $\mathbf{k}$-special-sound $(2\mu + 1)$-move interactive proof Π, with $\mathbf{k} = (k_1, \ldots, k_\mu)$. While the multi-round extractor has a natural recursive construction, it requires a more fine-grained analysis to show that it indeed implies knowledge soundness.

To avoid a cumbersome notation, in Section 6.1 we first handle $(2\mu+1)$-move interactive proofs in which the verifier samples all μ challenges uniformly at random from the *same* set $\mathcal{C}$. Subsequently, in Section 6.2, we argue that our techniques have a straightforward generalization to interactive proofs where the verifier samples its challenges from different challenge sets.

6.1 Multi-Round Protocols with a Single Challenge Set

Consider a deterministic dishonest Q-query random oracle prover $\mathcal{P}^*$, attacking the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of a $\mathbf{k}$-special-sound interactive proof Π on input x. We assume all challenges to be elements in the same set $\mathcal{C}$. Given a statement x as input, after making Q queries to the random oracle, $\mathcal{P}^*(x)$ outputs a proof $\pi = (a_1, \ldots, a_{\mu+1})$. We re-format the output and consider

$$I_1 := a_1 \,,\; I_2 := (a_1, a_2) \,, \ldots, \; I_\mu := (a_1, \ldots, a_\mu) \quad \text{and} \quad \pi$$

as $\mathcal{P}^*$'s output. Sometimes it will also be convenient to consider $I_{\mu+1} := (a_1, \ldots, a_{\mu+1})$. Furthermore, we extend $\mathcal{P}^*(x)$ to an algorithm $\mathcal{A}$ that additionally checks the correctness of the proof π. Formally, $\mathcal{A}$ runs

$\mathcal{P}^*(x)$ to obtain $\mathbf{I} = (I_1, \ldots, I_\mu)$ and π, queries RO to obtain $c_1 := \mathsf{RO}(I_1), \ldots, c_\mu := \mathsf{RO}(I_\mu)$, and then outputs

$$\mathbf{I}\,, \quad y := (a_1, c_1, \ldots, a_\mu, c_\mu, a_{\mu+1}) \qquad \text{and} \qquad v := V(x, y)\,,$$

where $V(x, y) = 1$ if y is an accepting transcript for the interactive proof Π on input x and $V(x, y) = 0$ otherwise.

Hence, $\mathcal{A}$ is a random-oracle algorithm making at most $Q + \mu$ queries (the queries done by $\mathcal{P}^*$, and the queries to $I_1, \ldots, I_\mu$). Moreover, $\mathcal{A}$ has a naturally defined success probability

$$\epsilon(\mathcal{A}) := \Pr\big(v = 1 : (I, y, v) \leftarrow \mathcal{A}^{\mathsf{RO}}\big)\,,$$

where $\mathsf{RO}\colon \{0,1\}^{\leq u} \to \mathcal{C}$ is distributed uniformly at random. As before, $\epsilon(\mathcal{A}) = \epsilon(\mathcal{P}^*, x)$.

Our goal is now to construct an extraction algorithm that, when given black-box access to $\mathcal{A}$, and thus to $\mathcal{P}^*$, aims to output a $\mathbf{k}$-tree of accepting transcripts (Definition 6). By the $\mathbf{k}$-special-soundness property of Π, a witness for statement x can be computed efficiently from these transcripts.

To this end, we recursively introduce a sequence of "subextractors" $\mathcal{E}_1, \ldots, \mathcal{E}_\mu$, where $\mathcal{E}_m$ aims to find a $(1, \ldots, 1, k_m, \ldots, k_\mu)$-tree of accepting transcripts. The main idea behind this recursion is that such a $(1, \ldots, 1, k_m, \ldots, k_\mu)$-tree of accepting transcripts is the composition of k_m appropriate $(1, \ldots, 1, k_{m+1}, \ldots, k_\mu)$-trees.

For technical reasons, we define the subextractors $\mathcal{E}_m$ as *random-oracle* algorithms, i.e., as algorithms that make $Q + \mu$ queries to a random oracle. As we will see, the recursive definition of $\mathcal{E}_m$ is very much like the extractor from the 3-move case, but with $\mathcal{A}$ replaced by the subextractor $\mathcal{E}_{m+1}$; however, for this to work we need the subextractor to be the same kind of object as $\mathcal{A}$, thus a random-oracle algorithm making the same number of queries. As base for the recursion, we consider the algorithm $\mathcal{A}$ (which outputs a single transcript, i.e., a $(1, \ldots, 1)$-tree); thus, the subextractor $\mathcal{E}_\mu$ (which outputs a $(1, \ldots, 1, k_\mu)$-tree) is essentially the extractor of the 3-move case, but with $\mathcal{A}$ now outputting an index *vector* $\mathbf{I}$, and with $\mathcal{E}_\mu$ being a *random-oracle* algorithm, so that we can then recursively replace the random-oracle algorithm $\mathcal{A}$ by $\mathcal{E}_\mu$ to obtain $\mathcal{E}_{\mu-1}$, etc.

Formally, the recursive definition of $\mathcal{E}_m$ from $\mathcal{E}_{m+1}$ is given in Figure 4, where $\mathcal{E}_{\mu+1}$ (the base case) is set to $\mathcal{E}_{\mu+1} := \mathcal{A}$, and where $\mathcal{E}_m$ exploits the following *early abort* feature of $\mathcal{E}_{m+1}$: like $\mathcal{A}$, the subextractor $\mathcal{E}_{m+1}$ computes the index vector it eventually outputs by running $\mathcal{P}^*(x)$ *as its first step* (see Lemma 7 below). This allows the executions of $\mathcal{E}_{m+1}$ in the repeat loop in Fig. 4 to abort after a single run of $\mathcal{P}^*$ if the requirement $I'_m = I_m$ on its index vector $\mathbf{I}$ is not satisfied, without proceeding to produce the remaining parts y', v' of the output (which would invoke more calls to $\mathcal{P}^*$).

The actual extractor $\mathcal{E}$ is then given by a run of $\mathcal{E}_1$, with the $Q + \mu$ random-oracle queries made by $\mathcal{E}_1$ being answered using lazy-sampling.

Remark 5. Let us emphasize that within *one* run of $\mathcal{E}_m$, except for the query to i for which the response is "reprogrammed", all the queries made by the multiple runs of the subextractor $\mathcal{E}_{m+1}$ in the repeat loop are answered *consistently*, both with the run of $\mathcal{E}_{m+1}$ in the first step and among the runs in the repeat loop. This means, a query to a value ξ that has been answered by η in a previous run on $\mathcal{E}_{m+1}$ (within the considered run of $\mathcal{E}_m$) is again answered by η, and a query to a value ξ' that has not been queried yet in a previous run on $\mathcal{E}_{m+1}$ (within the considered run of $\mathcal{E}_m$) is answered with a freshly chosen uniformly random $\eta' \in \mathcal{C}$. In *multiple* runs of $\mathcal{E}_m$, very naturally the random tape of $\mathcal{E}_m$ will be refreshed, and thus there is no guaranteed consistency among the answers to the query calls of $\mathcal{E}_{m+1}$ across multiple runs of $\mathcal{E}_m$.

The following lemma captures some technical property of the subextractors $\mathcal{E}_m$. Subsequently, Proposition 1 shows that $\mathcal{E}_m$, if successful, indeed outputs a $(1, \ldots, 1, k_m \ldots, k_\mu)$-tree of accepting transcripts. Proposition 2 bounds the success probability and expected run time of $\mathcal{E}_m$. All statements are understood to hold for any statement x and any $m \in \{1, \ldots, \mu + 1\}$.

Lemma 7 (Consistency of $\mathcal{P}^*$ and $\mathcal{E}_m$). *$\mathcal{E}_m$ obtains the index vector $\mathbf{I}$, which it eventually outputs, by running $(\mathbf{I}, \pi) \leftarrow \mathcal{P}^*(x)$ as its first step. In particular, for any random oracle RO, the index vector $\mathbf{I}$ output by $\mathcal{E}_m^{\mathsf{RO}}$ matches the one output by $\mathcal{P}^{*\mathsf{RO}}(x)$.*

Fig. 4. Subextractor $\mathcal{E}_m(x)$, as a $(Q+\mu)$-query random-oracle algorithm.

Parameters: $k, Q \in \mathbb{N}$
Black-box access to: $\mathcal{E}_{m+1}(x)$
Random oracle queries: $Q+\mu$

- Run $\mathcal{E}_{m+1}(x)$ as follows to obtain $(\mathbf{I}, y_1, v)$: relay the $Q+\mu$ queries to the random oracle and record all query-response pairs. Set $i := I_m$, and let c_i be the response to query i.
- If $v = 0$, abort with output $v = 0$.
- Else, repeat
  - sample $c'_i \in \mathcal{C} \setminus \{c_i\}$ (without replacement);
  - run $\mathcal{E}_{m+1}(x)$ as follows to obtain $(\mathbf{I}', y', v')$, aborting right after the initial run of $\mathcal{P}^*(x)$ if $I'_m \neq I_m$: answer the query to i with c'_i, while answering all other queries consistently if the query was performed by $\mathcal{E}_{m+1}(x)$ already on a previous run and with a fresh random value in $\mathcal{C}$ otherwise;

  until either $k_m - 1$ additional challenges c'_i with $v' = 1$ and $I'_m = I_m$ have been found or until all challenges $c'_i \in \mathcal{C}$ have been tried.
- In the former case, output $\mathbf{I}$, the k_m accepting $(1, \ldots, 1, k_{m+1}, \ldots, k_\mu)$-trees $y_1, \ldots, y_{k_m}$, and $v := 1$; in the latter case, output $v := 0$.

Proof. The first claim holds for $\mathcal{E}_{\mu+1} = \mathcal{A}$ by definition of $\mathcal{A}$, and it holds for $\mathcal{E}_m$ with $m \leq \mu$ by induction, given that $\mathcal{E}_m$ runs $\mathcal{E}_{m+1}$ as a first step. The claim on the matching index vectors then follows trivially. □

Proposition 1 (Correctness). *Let* RO *be an arbitrary random oracle, and let* $(\mathbf{I}, y_1, \ldots, y_{k_m}, v) \leftarrow \mathcal{E}_m^{\mathsf{RO}}(x)$. *If* $v = 1$ *then* $(y_1, \ldots, y_{k_m})$ *forms a* $(1, \ldots, 1, k_m, \ldots, k_\mu)$*-tree of accepting transcripts.*

Proof. All $k_{m+1} \cdots k_\mu$ transcripts in a $(1, \ldots, 1, k_{m+1}, \ldots, k_\mu)$-tree contain the same partial transcript $(a_1, c_1, \ldots, c_m, a_{m+1})$, i.e., the first $2m-1$ messages in all these transcripts coincide. Hence, any $(1, \ldots, 1, k_{m+1}, \ldots, k_\mu)$-tree of transcripts has a well-defined *trunk* $(a_1, c_1, \ldots, c_m, a_{m+1})$.

By induction on m, we will prove that if $v = 1$ then $(y_1, \ldots, y_{k_m})$ forms a $(1, \ldots, 1, k_m, \ldots, k_\mu)$-tree of accepting transcripts with trunk $(a_1, \mathsf{RO}(I_1), \ldots, \mathsf{RO}(I_m), a_{m+1})$, where $I_{m+1} = (a_1, \ldots, a_{m+1})$.

For the base case $m = \mu + 1$, recall that $\mathcal{E}_{\mu+1} = \mathcal{A}$, and that by definition of $\mathcal{A}$ and its output $(\mathbf{I}, y, v)$, if $v = 1$ then y is an accepting transcript, and thus a $(1, \ldots, 1)$-tree of accepting transcripts with $(a_1, \mathsf{RO}(I_1), \ldots, \mathsf{RO}(I_\mu), a_{\mu+1})$ as trunk where $I_{\mu+1} = (a_1, \ldots, a_{\mu+1})$, by definition of $\mathbf{I}$.

For the induction step, by the induction hypothesis on $\mathcal{E}_{m+1}$ and its output $(\mathbf{I}, y, v)$, if $v = 1$ then y is a $(1, \ldots, 1, k_{m+1}, \ldots, k_\mu)$-tree of accepting transcripts with trunk $(a_1, \mathsf{RO}(I_1), \ldots, a_m, \mathsf{RO}(I_m), a_{m+1})$, where $I_{m+1} = (a_1, \ldots, a_{m+1})$. This holds for $(\mathbf{I}, y_1, v)$ output by $\mathcal{E}_{m+1}$ in the first step of $\mathcal{E}_m$, but also for any invocation of $\mathcal{E}_{m+1}$ in the repeat loop with output $(\mathbf{I}', y', v')$, here with trunk $(a'_1, \mathsf{RO}'(I'_1), \ldots, a'_m, \mathsf{RO}'(I'_m), a'_{m+1})$, where $I'_{m+1} = (a'_1, \ldots, a'_{m+1})$ and RO' is such that $\mathsf{RO}'(I_j) = \mathsf{RO}(I_j)$ for all $j \neq m$, while $\mathsf{RO}(I_m) = c_i$ and $\mathsf{RO}'(I_m) = c'_i$. By definition of the output of $\mathcal{E}_m$, for y_1 and y' occurring in the output of $\mathcal{E}_m$, it is ensured that $I_m = I'_m$.

Now note that, by Lemma 7, for the purpose of the argument, $\mathcal{E}_m$ could have run $\mathcal{P}^*$ instead of $\mathcal{E}_{m+1}$ to obtain $\mathbf{I}$ and $\mathbf{I}'$. Therefore, by definition of the index vectors outputted by $\mathcal{P}^*$, which is such that I_j is a (fixed-size) prefix of I_m for $j < m$, it follows that also $I_j = I'_j$ for all $j < m$.

Therefore, the output $y_1, \ldots, y_{k_m}$ of $\mathcal{E}_m$ forms a $(1, \ldots, 1, k_m, \ldots, k_\mu)$-tree of accepting transcripts with trunk $(a_1, \mathsf{RO}(I_1), \ldots, a_{m-1}, \mathsf{RO}(I_{m-1}), a_m)$, where $I_m = (a_1, \ldots, a_m)$. This completes the proof. □

Proposition 2 (Run Time and Success Probability). *Let* $K_m = k_m \cdots k_\mu$. *The extractor* $\mathcal{E}_m$ *makes an expected number of at most* $K_m + Q \cdot (K_m - 1)$ *queries to* $\mathcal{A}$ *(and thus to* $\mathcal{P}^*$ *and successfully outputs* $v = 1$ *with probability at least*

$$\frac{\epsilon(\mathcal{A}) - (Q+1) \cdot \kappa_m}{1 - \kappa_m}$$

where $\kappa_m := \mathrm{Er}\big((k_m, \ldots, k_\mu); N\big)$ *is as defined in Equation 1.*

Proof. The proof goes by induction on m. The base case $m = \mu + 1$ holds trivially, understanding that $K_{\mu+1} = 1$ and $\mathrm{Er}(\emptyset, N) = 0$. Indeed, $\mathcal{E}_{\mu+1}$ makes 1 call to $\mathcal{A}$ and outputs $v = 1$ with probability $\epsilon(\mathcal{A})$. Alternatively, we can take $m = \mu$ as base case, which follows immediately from Lemma 4.

For the induction step, we assume now the lemma is true for $m' = m + 1$ and consider the extractor $\mathcal{E}_m$. As in the 3-move case, we observe that, within a run of $\mathcal{E}_m$, all the queries that are made by the different invocations of $\mathcal{E}_{m+1}$ are answered *consistently* using lazy sampling, except for the queries to the index i, which is answered with different responses c'_i. This is indistinguishable from having them answered by a full-fledged random oracle $\mathbf{c} \in \mathcal{C}^U$, where we recall that $U = |\{0,1\}^{\leq u}|$ and therefore a full-fledged random oracle corresponds to a vector $\mathbf{c} \in \mathcal{C}^U$ encoding its function table. Thus, the extractor is actually running the abstract sampling game from Figure 2.

However, in contrast to the instantiation of Section 4, the entries of the array M are now *probabilistic*. Namely, while $\mathcal{A}$ is deterministic, the extractor $\mathcal{E}_{m+1}$ is a probabilistic algorithm. Fortunately, this does not influence the key properties of the abstract sampling game. For the purpose of the analysis we may namely fix the randomness of the extractor $\mathcal{E}_{m+1}$. By linearity of the success probability and the expected run time, the bounds that hold for any fixed choice of randomness also hold when averaged over the randomness. Thus, we can apply Lemma 2 and Lemma 5 to obtain bounds on the success probability and the expected run time.[5]

In order to control the parameters P and T, which occur in the bounds of these lemmas, we make the following observation. A similar observation was required in the proof of Lemma 4.

First, by Lemma 7, the index vector $\mathbf{I}$ outputted by $\mathcal{E}_{m+1}$ matches the index vector outputted by $\mathcal{P}^*$, when given the same random oracle $\mathbf{c} \in \mathcal{C}^U$. Second, since $\mathcal{P}^*$ is deterministic, its output can only change when the random oracle is reprogrammed at one of the indices $i \in \{0,1\}^{\leq u}$ queried by $\mathcal{P}^*$. Therefore, for every random oracle $\mathbf{c} \in \mathcal{C}^U$, there exists a subset $S(\mathbf{c}) \subset \{0,1\}^{\leq u}$ (indicating the queries made by $\mathcal{P}^*$) such that $\mathcal{P}^*$'s output stays the same when the random oracle is reprogrammed at an index $i \notin S(\mathbf{c})$. In particular, $I_j(\mathbf{c}) = I_j(\mathbf{c}')$ for all $1 \leq j \leq \mu$ and $\mathbf{c}, \mathbf{c}' \in \mathcal{C}^U$ with $c_i = c'_i$ for all $i \in S(\mathbf{c})$. Hence, the conditions of Lemma 3 and Lemma 6 are satisfied, and it follows that $P \leq Q + 1$ and $T \leq Q$. We are now ready to analyze the success probability and the expected number of $\mathcal{A}$ queries of $\mathcal{E}_m$.

Success Probability. By the induction hypothesis, the success probability p_{m+1} of $\mathcal{E}_{m+1}$ is bounded by

$$p_{m+1} \geq \frac{\epsilon(\mathcal{A}) - (Q+1) \cdot \kappa_{m+1}}{1 - \kappa_{m+1}} \,.$$

Then, by Lemma 2 and Lemma 3, the success probability of $\mathcal{E}_m$ is bounded by

$$p_m \geq \frac{N}{N - k_m + 1} \left(p_{m+1} - (Q+1)\frac{k_m - 1}{N} \right) \geq \frac{N}{N - k_m + 1} \left(\frac{\epsilon(\mathcal{A}) - (Q+1) \cdot \kappa_{m+1}}{1 - \kappa_{m+1}} - (Q+1)\frac{k_m - 1}{N} \right).$$

By the recursive property (2) of $\kappa_m := \mathrm{Er}\big(k_m, \ldots, k_\mu; N, \ldots, N\big)$, it follows that

$$\frac{N - k_m + 1}{N}(1 - \kappa_{m+1}) = 1 - \kappa_m \,.$$

[5] To be more precise, to allow for fresh randomness in the different runs of $\mathcal{E}_{m+1}$ within $\mathcal{E}_m$, we first replace the randomness of $\mathcal{E}_{m+1}$ by $F(\mathbf{c})$ for a random function F, where $\mathbf{c} \in \mathcal{C}^U$ is the random oracle providing the answers to $\mathcal{E}_{m+1}$'s queries, and then we fix the choice of F and average over F after having applied Lemma 2 and Lemma 5.

Hence,

$$
\begin{aligned}
p_m &\geq \frac{\epsilon(\mathcal{A}) - (Q+1)\cdot\kappa_{m+1}}{1-\kappa_m} - (Q+1)\frac{k_m - 1}{N - k_m + 1} \\
&= \frac{1}{1-\kappa_m}\left(\epsilon(\mathcal{A}) - (Q+1)\cdot\left(\kappa_{m+1} + (1-\kappa_m)\frac{k_m-1}{N-k_m+1}\right)\right) \\
&= \frac{1}{1-\kappa_m}\left(\epsilon(\mathcal{A}) - (Q+1)\cdot\left(1 - (1-\kappa_m)\cdot\frac{N}{N-k_m+1} + (1-\kappa_m)\frac{k_m-1}{N-k_m+1}\right)\right) \\
&= \frac{\epsilon(\mathcal{A}) - (Q+1)\cdot\kappa_m}{1-\kappa_m}\,,
\end{aligned}
$$

which proves the claimed success probability.

Expected Number of $\mathcal{A}$-Queries. Let the random variable T_m denote the number of $\mathcal{A}$-queries made by extractor $\mathcal{E}_m$. By the induction hypothesis, it holds that

$$\mathbb{E}[T_{m+1}] \leq K_{m+1} + Q(K_{m+1} - 1)\,.$$

We make one crucial observation, allowing us to achieve the claimed query complexity, linear in Q. Namely, we can view the run of a (sub)extractor as a *two-stage* algorithm that allows an *early abort*. By Lemma 7, after only one invocation of $\mathcal{A}$-query $\mathcal{E}_{m+1}$ already returns the index I_m. At this stage, $\mathcal{E}_m$ can decide whether to continue the execution of $\mathcal{E}_{m+1}$ or to *early abort* this execution. If the index is incorrect, i.e., it does not match the one obtained in the first invocation of $\mathcal{E}_{m+1}$, then $\mathcal{E}_m$ early aborts the execution of $\mathcal{E}_{m+1}$. Only if the index is correct, the $\mathcal{E}_{m+1}$ execution has to be finished.

For this reason, we define the function $\mathbf{c} \mapsto \Gamma(\mathbf{c})$, where $\Gamma(\mathbf{c})$ is the (expected) costs of running $\mathcal{E}_{m+1}$ (completely) with random oracle $\mathbf{c} \in \mathcal{C}^U$. Moreover, we set $\gamma = 1$ indicating the cost of an early abort invocation of $\mathcal{E}_{m+1}$. These cost functions measure the expected number of calls to $\mathcal{A}$.

Hence, by Lemma 5 and Lemma 6, the expected cost of running $\mathcal{E}_m$ is at most

$$
\begin{aligned}
\mathbb{E}[T_m] &\leq k_m \cdot \mathbb{E}[\Gamma(C)] + \gamma\cdot Q\cdot(k_m - 1) = k_m\cdot\mathbb{E}[T_{m+1}] + Q\cdot(k_m-1) \\
&\leq K_m + Q\cdot(K_m - k_m) + Q\cdot(k_m - 1) = K_m + Q\cdot(K_m - 1)\,,
\end{aligned}
$$

where C is distributed uniformly at random in $\mathcal{C}^U$. This completes the proof of the proposition.

□

The existence of extractor $\mathcal{E}_1$, combined with the $\mathbf{k}$-special-soundness property, implies the following.

Theorem 2 (FS Transformation of a $(k_1,\ldots,k_\mu)$-Special-Sound Protocol). *The Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of a $\mathbf{k} = (k_1,\ldots,k_\mu)$-special-sound interactive proof Π, in which all challenges are sampled from a set $\mathcal{C}$ of size N, is knowledge sound with knowledge error*

$$\kappa_{\mathrm{fs}}(Q) = (Q+1)\,\mathrm{Er}(\mathbf{k}; N)\,.$$

where $\mathrm{Er}(\mathbf{k}; N)$ is the knowledge error of the interactive proof Π.

6.2 Multi-Round Protocols with Arbitrary Challenge Sets

Thus far, we considered and analyzed multi-round interactive proofs in which all challenges are sampled uniformly at random from the *same* set $\mathcal{C}$ of cardinality N. However, it is straightforward to verify that our techniques also apply to multi-round interactive proofs with different challenge sets, i.e., where the i-th challenge is sampled from a set $\mathcal{C}_i$ of cardinality N_i.

A natural first step in this generalization is to consider μ random oracles $\mathsf{RO}_i\colon \{0,1\}^{\leq u} \to \mathcal{C}_i$ instead of one. Besides some additional bookkeeping, all the reasoning goes through unchanged. Indeed, everything works as is when the prover $\mathcal{P}^*$ has the additional freedom to choose which random oracle it queries. Thus, we obtain the following generalization of Theorem 2.

Theorem 3 (FS Transformation of a k-out-of-N Special-Sound Interactive Proof). *The Fiat-Shamir transformation of a* $\mathbf{k}$*-out-of-*$\mathbf{N}$ *special-sound interactive proof* $(\mathcal{P},\mathcal{V})$ *is knowledge sound with knowledge error*

$$\kappa_{\mathrm{fs}}(Q) = (Q+1)\,\mathrm{Er}(\mathbf{k};\mathbf{N})\,.$$

where $\mathrm{Er}(\mathbf{k},\mathbf{N})$ *is the knowledge error of the interactive proof* $(\mathcal{P},\mathcal{V})$*.*

Remark 6. Alternatively, one could fix μ mappings $f_i\colon \{0,1\}^* \to \mathcal{C}_i$ and define the random oracle to output sufficiently long bit-strings. As before, this allows the prover $\mathcal{P}^*(x)$ to take as input a single random oracle. Of course, this approach closely resembles practice, where the random oracles are replaced hash functions. However, one must be careful, since distinct bit-strings do not necessarily map to distinct challenges and uniformly random bit-strings do not necessary correspond to uniformly random challenges.

7 The Fiat-Shamir Transformation of Parallel Repetitions

In the previous sections we have established a positive result; for a broad class of interactive proofs the Fiat-Shamir security loss is only linear in the query complexity Q and independent of the number of rounds. One might therefore wonder whether the generic $(Q+1)^\mu$ security loss, for $(2\mu+1)$-move protocols, is only tight for contrived examples. In this section, we show that this is *not* the case. We demonstrate a non-trivial attack on the Fiat–Shamir transformation of the parallel repetition of $\mathbf{k}$-special-sound protocols.

Let $\Pi = (\mathcal{P},\mathcal{V})$ be a $(2\mu+1)$-move $\mathbf{k}$-special-sound interactive proof. By $\Pi^t = (\mathcal{P}^t,\mathcal{V}^t)$ we denote its *t-fold parallel repetition*. That is, the prover $\mathcal{P}^t(x;w)$ runs t instances of $\mathcal{P}(x;w)$, i.e., each message is a tuple $(a^1,\ldots,a^t)$ of messages, one for each parallel thread of execution. Likewise, the verifier $\mathcal{V}^t(x)$ runs t instances of $\mathcal{V}(x)$ in parallel, i.e., each challenge is a tuple $(c^1,\ldots,c^t)$ of challenges, one for each parallel thread of the execution. The verifier accepts if all parallel instances are accepting.

Our result. Assuming certain natural properties on Π, which are satisfied by typical examples, and assuming again for simplicity that the challenge spaces $\mathcal{C}_i$ all have the same cardinality N, we show that there exists a malicious Q-query prover $\mathcal{P}^*$, attacking $\mathsf{FS}[\Pi^t]$, that (for any statement x) succeeds in convincing the verifier with probability at least $1/2\cdot Q^\mu \mu^{-t-\mu}\cdot \mathrm{Er}(\mathbf{k};N)^t$, where $\mathrm{Er}(\mathbf{k};N)$ is the knowledge error of Π and $\mathrm{Er}(\mathbf{k};N)^t$ is the knowledge error of Π^t.

Proposition 3. *Let* $k,\mu\in\mathbb{N}$ *and let* $t,Q\in\mathbb{N}$ *be integer multiples of* μ *such that* $Q\cdot\left(\frac{k-1}{N}\right)^{t/\mu}\leq 1/4$*. There are (natural)* $(2\mu+1)$*-move* $(k,\ldots,k)$*-out-of-*$(N,\ldots,N)$ *special-sound interactive proofs* Π *for which there exists a* Q*-query dishonest prover* $\mathcal{P}^*$ *against* $(\mathcal{P},\mathcal{V}) = \mathsf{FS}[\Pi^t]$ *such that, for any statement* $x\in\{0,1\}^*$*,*

$$\epsilon(\mathcal{P}^*,x) = \Pr\left(\mathcal{V}^{\mathsf{RO}}(x,\mathcal{P}^{*,\mathsf{RO}}(x))=1\right) \geq \left(1-\left(1-(\tfrac{k-1}{N})^{t/\mu}\right)^{Q/\mu}\right)^\mu \geq \frac{1}{2}\left(\frac{Q}{\mu}\right)^\mu\left(\frac{\mathrm{Er}(\mathbf{k};N)}{\mu}\right)^t\,,$$

where $\mathbf{k}=(k,\ldots,k)$*.*

Proposition 3 shows that the security loss of the Fiat-Shamir transformation, when applied to the t-fold parallel repetition Π^t of Π, is at least $1/2\cdot Q^\mu\cdot\mu^{-t-\mu}$. This stands in stark contrast to a single execution of a $\mathbf{k}$-special-sound protocol, where the loss is linear in Q and independent of μ. We go on to discuss the inner workings of our attack. Note that, for simplicity, above and below we restrict our attention to $\mathbf{k}=(k,\ldots,k)$ and assume $t=\mu\cdot t'$ and $Q=\mu\cdot Q'$ to be multiples of μ. With some adjustments to the bound and the reasoning, these assumptions can be avoided.

Special-Unsoundness. Let $\boldsymbol{\ell} = (\ell, \ldots, \ell)$ where $\ell \leq k-1$. The attack on $\mathsf{FS}[\Pi^t]$ uses a property most $\mathbf{k}$-special-sound protocols Π satisfy, namely that there exists an efficient attack strategy $\mathcal{A}$ against Π which tries to guess challenges up front so that:

- In any round, $\mathcal{A}$ can prepare and send a message so that if he is lucky and the next challenge is within ℓ out of the N possible choices, $\mathcal{A}$ will be able to complete the protocol and have the verifier accept;
- Until $\mathcal{A}$ was lucky with the challenge, it sends high-entropy messages.

We call protocols which suffer from such an attack strategy $\boldsymbol{\ell}$*-special-unsound*. The last point ensures that for the FS transformation, an attacker can produce many different challenges for a fixed round (and try to be lucky many times). These requirements are very common and often satisfied. For example, the folding technique of [BCC+16], when used to fold two parts into one, satisfies $(3, \ldots, 3)$-special-soundness and $(2, \ldots, 2)$-special-unsoundness. Note here, that while the *honest* prover in [BCC+16] is *deterministic*, a dishonest prover can still produce randomized messages (and hope to be lucky with the received challenge).

Attack on $\mathsf{FS}[\Pi^t]$. The basic idea of the attack is that (groups of) parallel threads can be attacked individually and independently from each other over the different rounds of the protocol. Concretely, the attack is given by the adversary $\mathcal{P}^*$ against $\mathsf{FS}[\Pi^t]$, which makes up to $Q = \mu \cdot Q'$ queries, defined as follows: $\mathcal{P}^*$ runs attack strategy $\mathcal{A}$ in parallel against all $t = \mu \cdot t'$ threads. Let us call a thread *green* if strategy $\mathcal{A}$ succeeds in guessing the challenge for that thread (and hence, $\mathcal{V}$ will eventually accept for that thread). Otherwise, a thread is *red*. All threads start out red, and the goal of $\mathcal{P}^*$ is to turn all threads green. To do so, in every round $\mathcal{P}^*$ tries to turn at least $t' = t/\mu$ red threads into green threads, or all red threads into green threads (if fewer than t/μ remain). For this, $\mathcal{P}^*$ uses $\mathcal{A}$ to get the messages which it feeds to the random oracle. If $\mathcal{P}^*$ was lucky with the received challenges for at least $t' = t/\mu$ threads, then enough red threads turn green. Else, $\mathcal{P}^*$ tries the considered round again. (Here we use that $\mathcal{A}$ has entropy in its messages, so that we get fresh challenges from the random oracle). The dishonest prover $\mathcal{P}^*$ tries up to $Q' = Q/\mu$ times per round until it gives up (and fails).

The number of queries $\mathcal{P}^*$ makes to the random oracle is at most Q, hence $\mathcal{P}^*$ is a Q-query adversary. The probability that $\mathcal{P}^*$ succeeds for any try in any round is at least $(\frac{\ell}{N})^{t/\mu}$. Therefore, since the $\mathcal{P}^*$ makes at most Q/μ queries in every round, the success probability for any fixed round is at least

$$1 - \left(1 - \left(\frac{\ell}{N}\right)^{t/\mu}\right)^{Q/\mu} . \tag{11}$$

Now observe that, for all $n \in \mathbb{N}$ and $x \geq 0$ with $nx \leq 1/2$,

$$|1 - (1-x)^n - nx| = \left| -\sum_{i=2}^{n} (-1)^i \binom{n}{i} x^i \right| \leq \sum_{i=2}^{\infty} (nx)^i \leq \frac{(nx)^2}{1-nx} \leq 2(nx)^2 ,$$

Let us now assume that $\frac{Q}{\mu}\left(\frac{\ell}{N}\right)^{t/\mu} \leq Q \cdot \left(\frac{\ell}{N}\right)^{t/\mu} \leq 1/4$. Then, applying this observation to Equation 11 shows that in each round $\mathcal{P}^*$ succeeds with probability at least

$$\frac{Q}{\mu}\left(\frac{\ell}{N}\right)^{t/\mu} - 2\left(\frac{Q}{\mu}\right)^2 \left(\frac{\ell}{N}\right)^{2t/\mu} = \frac{Q}{\mu}\left(\frac{\ell}{N}\right)^{t/\mu}\left(1 - 2\frac{Q}{\mu}\left(\frac{\ell}{N}\right)^{t/\mu}\right) .$$

Hence, $\mathcal{P}^*$ succeeds (in all μ rounds) with probability at least

$$\left(\frac{Q}{\mu}\right)^{\mu}\left(\frac{\ell}{N}\right)^{t}\left(1 - 2\frac{Q}{\mu}\left(\frac{\ell}{N}\right)^{t/\mu}\right)^{\mu} \geq \left(\frac{Q}{\mu}\right)^{\mu}\left(\frac{\ell}{N}\right)^{t}\left(1 - 2Q\left(\frac{\ell}{N}\right)^{t/\mu}\right) ,$$

where we use that $(1-z)^n \geq 1-nz$ for all $n \in \mathbb{N}$ and $z \in [0,1]$. Since we assume that $Q \cdot \left(\frac{\ell}{N}\right)^{t/\mu} \leq 1/4$, it follows that $\mathcal{P}^*$ succeeds with probability at least

$$\frac{1}{2}\left(\frac{Q}{\mu}\right)^{\mu}\left(\frac{\ell}{N}\right)^{t}.$$

To complete the analysis of $\mathcal{P}^*$'s success probability, we observe that, if $\ell = k-1$ (as assumed),

$$\mathrm{Er}(\mathbf{k}; N) = 1 - \left(1 - \frac{k-1}{N}\right)^{\mu} \leq \mu \cdot \frac{k-1}{N} = \mu \cdot \frac{\ell}{N}.$$

Hence, the success probability of $\mathcal{P}^*$ is at least

$$\frac{1}{2}\left(\frac{Q}{\mu}\right)^{\mu}\left(\frac{\mathrm{Er}(\mathbf{k}; N)}{\mu}\right)^{t}.$$

This shows that the security loss of the Fiat-Shamir transformation, when applied to Π^t, is at least $1/2 \cdot Q^{\mu} \cdot \mu^{-t-\mu}$. This result is summarized in Proposition 3. Recall that we assume t and Q to be divisible by μ. In general, i.e., when dropping this assumption, the success probability has lower bound $1/2 \cdot \lfloor Q/\mu \rfloor^{\mu} \cdot (\mathrm{Er}(\mathbf{k}; N)/\mu)^{\lceil t/\mu \rceil \mu}$.

8 Acknowledgments

The first author was supported by EU H2020 project No. 780701 (PROMETHEUS) and the Vraaggestuurd Programma Cyber Security & Resilience, part of the Dutch Top Sector High Tech Systems and Materials program. The third author was supported by the topic Engineering Secure Systems (46.23.01) of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

References

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In *CRYPTO'21*, volume 12826 of *LNCS*, pages 549–579. Springer, 2021.

AF21. Thomas Attema and Serge Fehr. Parallel repetition of $(k_1, \ldots, k_\mu)$-special-sound multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1259, 2021. `https://eprint.iacr.org/2021/1259`.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

BL02. Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th ACM STOC*, pages 484–493. ACM Press, May 2002.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

GT21. Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In *CRYPTO'21*, volume 12827 of *LNCS*, pages 64–93. Springer, 2021.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

Wik18. Douglas Wikström. Special soundness revisited. Cryptology ePrint Archive, Report 2018/1157, 2018. `https://eprint.iacr.org/2018/1157`.

Wik21. Douglas Wikström. Special soundness in the random oracle modeld. Cryptology ePrint Archive, Report 2021/1265, 2021. `https://eprint.iacr.org/2021/1265`.