Simulation Security in the Random Oracle Model

Jérémi Do Dinh

School of Computer and Communication Sciences

Master's Thesis

August 2024

Responsible
Prof. Alessandro Chiesa
EPFL / COMPSEC

Supervisor
Giacomo Fenzi
EPFL / COMPSEC

Abstract

Cryptographic arguments are versatile and can be used to construct cryptographic primitives, such as signature schemes and adaptively secure encryption schemes. Compared to the classical notions of zero-knowledge, soundness, and knowledge soundness, these constructions require a strengthening of the security properties required by the underlying argument. In particular, the (strictly) stronger notions of simulation soundness and simulation knowledge soundness are necessary.

In this work, we consider arguments in the pure Random Oracle Model (ROM), formally defining the notions of simulation security required for the constructions we analyze and providing security reductions with concrete security bounds. The frameworks used for the constructions are not novel; however, prior work either differs in the cryptographic model used or analyzes them with asymptotic security notions.

"What you call passion is not a spiritual force, but friction between the soul and the outside world. Where passion dominates, that does not signify the presence of greater desire and ambition, but rather the misdirection of these qualities toward and isolated and false goal, with a consequent tension and sultriness in the atmosphere. Those who direct the maximum force of their desires toward the center, toward true being, toward perfection, seem quieter than the passionate souls because the flame of their fervor cannot always be seen. In argument, for example, they will not shout or wave their arms. But I assure you, they are nevertheless burning with subdued fires."

— Hermann Hesse, *The Glass Bead Game*

Acknowledgements.

I want to express my sincere gratitude to Alessandro Chiesa and Giacomo Fenzi for their thorough supervision and for giving me the opportunity to complete my Master's thesis within the group. My heartfelt thanks go out to everyone at Compsec for making me feel welcome in the lab throughout these months. I give special thanks to Burcu for her kindness and supportive conversations.

I am deeply grateful to my family. Mama and Papa, thank you teaching me how to love, for always being here, and opening so many doors for me. Mikuś, I'm glad I have a brother I can call my best friend. You're probably the coolest person I know. Never change.

I want to thank my best friend, Nidal, for being like a big brother to me. I am also grateful to the rest of the Montreal gang — Jaafar "Le Sumulateur", Pat, Fia, and Sean — for helping me see a different side of life and maintain balance throughout these years.

This journey at EPFL was made enjoyable thanks to the fantastic friends I've made: Pietro, Andrea, Lorenzo, Dũng, Robert, Alex, Mark and many more. I'm also thankful to Grześ and Konrad for their great company and engaging discussions. To the friends I've met along the way — Luca, Gennaro, Tomek, Charly, Olympe, Chloé, Julie, Viktor, Adam, Nathan, Luke — that I'm so glad to have in my life.

Let the music play. It's time for Discoplay.

Contents

1 Introduction

A popular tool to study cryptographic protocols is the *random oracle model* (abbreviated ROM), where each party (honest or malicious) taking part in a protocol receives query access to the random function, known as the *random oracle*. Each query to the random oracle returns an entirely random fixed-length output. [BR93] first advocated the utility of random oracles for cryptographic protocols.

The ROM presents both advantages and limitations. Cryptographic hash functions typically replace the idealized random oracle, as hash functions aim to be efficient realizations of the latter. Although [CGH04] demonstrated the theoretical limitations of this, it remains a prevalent approach in modern cryptographic practice.

Quite notably, using a hash function eliminates the need for any trusted setup ceremonies, which are often required in, for instance, the Common Reference String (CRS) model. In our scenario, such a setup boils down to a simple agreement between parties on which hash function to use. The transparency of the setup has enabled rapid adoption in applications where delegation of computation is practical.

Furthermore, hash functions are crucial in rendering interactive proof systems non-interactive. The most notable methods for this are the Fiat-Shamir transformation [FS86] for sigma protocols [Sch90] and interactive-proofs [GMR85; Bab85], the Micali transformation [Mic00] for probabilistically checkable proofs (PCPs) [BFLS91], and the BCS transformation for interactive oracle proofs (IOPs) [BCS16]. In such scenarios, hash functions bridge information-theoretic constructions and practical realizations, with the random oracle model providing an intuitive framework for analyzing security.

Hash functions undergo strenuous standardization processes [Nat]. Security usually holds when protocols are instantiated with one, albeit on a heuristic basis. It is also worth noting that many cryptographic constructions studied in the ROM have been shown to maintain security [DFMS19; LZ19; CMS19] when extended to its quantum counterpart (the QROM). While this implication does not generally hold [BDFLSZ11; YZ22], these results demonstrate that the ROM is instrumental in advances towards a post-quantum world.

The recently published book "Building Cryptographic Proofs from Hash Functions", by Alessandro Chiesa and Eylon Yogev [CY24], rigorously formalizes the landscape of cryptographic arguments in the ROM. The book provides a comprehensive treatment of fundamental cryptographic notions focusing on the *pure* random oracle model: the case where an adversary is bounded only by the number of queries he makes to the random oracle. In particular, [CY24, Chapter 5] defines the notions of zero-knowledge [GMR85] and knowledge soundness [GMR85; FS89; BG93] in the context of non-interactive arguments in the ROM, along with a rigorous exposition of the parameters upon which the error terms depend.

In this work, we study stronger notions of security in the random oracle model. Namely, we study simulation security. The two central notions that fall under this term, which we define and study, are *simulation soundness* and *simulation knowledge soundness* (also called *simulation extractability*).

1.1 Our Results

To begin with, we generalize the notion of adaptive non-interactive zero-knowledge in the random oracle model from [CY24] to provide security against an adversary capable of seeing multiple argument strings for *true* statements of his choice. We follow this with a discussion on simulation security, which includes formal definitions of the notions of simulation soundness and simulation

knowledge soundness in the random oracle model with concrete security parameters. Similarly to [DHLW10], we make a distinction between two overarching types of simulation security: *any* and *true* simulation security. In the former, the adversary can query the random oracle on *any* instances, including false ones. In the latter, the adversary can only see simulated argument strings for *true* instances.

In this work, we leverage Non-Interactive Arguments (NARGs) with specific properties to formalize the construction of two fundamental cryptographic primitives in the random oracle model. Specifically, we present: (1) A signature scheme satisfying existential unforgeability under chosen message attack (EUF-CMA); and (2) An encryption scheme secure under chosen-ciphertext attacks (CCA-2). For both constructions, we employ zero-knowledge NARGs with carefully selected simulation security properties in a black-box manner. This approach allows us to achieve the desired security guarantees while maintaining a clear and modular design.

The signature scheme is constructed using a framework first introduced in [KV09] based on hard relations (a generalization of one-way functions) and a *true*-simulation extractable zero-knowledge NARG. The encryption scheme follows the Naor-Young paradigm [NY90], combining a CPA secure encryption scheme with a *true*-simulation sound zero-knowledge NARG. Most importantly, we show that no construction in this work requires *any*-simulation security.

Concrete security. The original formalizations of provable security, as introduced in [GM84], are asymptotic: security bounds are negligible functions of a security parameter. Having this type of guarantee does not always help when it comes to instantiating a primitive in practice: internally, there may be other parameters in a given cryptographic construction, which can significantly affect the security, but that polynomial equivalence does not capture.

In every definition and construction in this work, we follow the approach from [BKR94; BGR95; BDJR97], wherein all of the resources given to an adversary are parameters upon which the security bound depends. The proofs of security of the constructions consider the overhead for each parameter involved in the security reduction. As such, the security bounds are, therefore, non-asymptotic: they are functions of individual parameters.

1.2 Related Work

The study of simulation security emerged in a line of cryptographic research concerned with non-malleability. *Non-malleable cryptography* was first introduced in the work of Dolev, Dwork, and Naor [DDN91; DDN00], who presented constructions for non-malleable zero-knowledge and non-malleable commitments in the plain standard model. This work also presented the first known CCA-2 secure encryption scheme. The work of Sahai [Sah99] (and subsequently [DDOPS01]) strengthened the notion of non-malleability by introducing and defining *simulation soundness* in the common reference string (CRS) model. [Sah99] also introduced a much simplified construction of a CCA-2 secure encryption scheme. *Simulation extractability* informally states that an adversary cannot prove statements for which he does not know a witness, even if he has access to a simulation oracle. It encompasses the properties necessary to achieve, for instance, security in the universal composability (UC) framework and is considered the strongest of the three [PR05; DDOPS01; GMY06; JP11; FKMV12]. All of these provide, on different levels, guarantees against malleability attacks.

A formal study of the relationships between the notions of non-malleability [DDN91; DDN00], simulation soundness [Sah99], and simulation extractability [PR05] was not conducted until [JP11]. While simulation soundness immediately implies non-malleability, the converse is true when the

non-malleable zero-knowledge protocol (as defined in [DDN91; DDN00]) is also an argument of knowledge. Most surprisingly, [JP11] show that simulation extractability, often believed to be the strongest of the three notions, does not necessarily imply simulation soundness. We note that their analysis is in the plain standard model, and the proof is for the non-adaptive case, with the adaptive case left as an open question. They also acknowledge that the presence of their counter-example could be related to the lack of consistency between the formulations of the definitions of simulation soundness and simulation extractability (from [Sah99] and [PR05], respectively). In particular, they do not dismiss the possibility of independent, intuitive, and compatible formulations of the notions, conceptually capturing the ideas.

In this work, we provide such intuitive and compatible definitions. Briefly, for simulation soundness, our definition states that no adversary can provide an accepting argument string for a false statement despite having access to a bounded number of simulated argument strings, even for false statements. Simulation knowledge soundness is a strengthening of this, which additionally requires that an efficient extractor can compute a witness for any statement the adversary can prove, even after seeing a bounded number of simulated argument strings. We may think of simulation soundness as a case of simulation knowledge soundness where the extractor is all powerful: since a witness does not exist, no extraction algorithm, no matter how resourceful, will be able to compute a witness.

Dodis et al. [DHLW10] were the first to make the distinction between *any* and *true* simulation security and show that the full-fledged notion of simulation security, where the adversary may query the simulation oracle on false instances, is not necessary for constructing encryption and signature schemes. However, their analysis is in the CRS model and is limited to asymptotic security bounds.

The work by Faust et al. [FKMV12] began exploring the properties above outside the Common Reference String (CRS) model. They demonstrated that the non-malleability of Σ-protocols is preserved when these protocols are rendered non-interactive via the Fiat-Shamir transform [FS86]. Notably, their definitions of simulation security are tailored to the Random Oracle Model (ROM). However, the analysis is also limited to asymptotic security notions.

Our work extends this approach by considering how the reductions affect the parameters concretely and, therefore, steps into the realm of concrete security. This additional effort provides a more comprehensive understanding of the schemes' real-world security implications and facilitates the choice of parameters in deployed scenarios.

Building upon [CY24], this work expands the scope of definitions and strives to formalize simulation security notions with comparable rigor and precision.

2 Preliminaries

This section defines basic security notions for non-interactive arguments in the random oracle model. The definitions are taken from [CY24] directly, except Definitions 2.8 and 2.9: the first is a generalization of the zero-knowledge definition from [CY24, Chapter 5] and the second is a computational relaxation of this generalization. The definitions presented here are foundational for the subsequent section on simulation security (Section 3), the contents of which serve as building blocks for the constructions presented in Sections 4 and 5.

All the definitions in this work are *adaptive*, where, in a given security game, an adversary interacts with the random oracle and chooses an instance (or instances) based on this interaction. Adaptive definitions contrast the weaker non-adaptive definitions, where the adversary is bound to a particular instance chosen prior to his execution. [CY24, Chapter 4.1] includes a complete discussion of the differences between the two notions.

The section is structured as follows.

- In Section 2.1, we clarify the notation used throughout this work.
- In Section 2.2, we define the *random oracle model*.
- In Section 2.3, we define *non-interactive arguments* in the ROM and related security notions.
- In Section 2.4, we define *knowledge soundness*.
- In Section 2.5, we define *zero-knowledge*.

2.1 Notation

Randomized algorithms. We use the notation $y \leftarrow A(x)$ to denote that y results from running a randomized algorithm A on input x. We write $A(\,\cdot\,;\rho)$ to make explicit the randomness ρ used throughout the execution of A.

Oracles. We use the notation $y \xleftarrow{\mathsf{tr}} A^{\mathcal{O}}(x)$ to denote an algorithm A outputting y on input x, given query access to an oracle $\mathcal{O}$, where tr is the query-answer trace of A's interaction with $\mathcal{O}$. If A has access to multiple oracles, we identify the traces with appropriate subscripts (e.g., $y \xleftarrow{\mathsf{tr}_{\mathcal{O}_1},\mathsf{tr}_{\mathcal{O}_2},\cdots} A^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x)$). An adversary with access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n$ is $(t_1, t_2, \ldots, t_n)$-query admissible if for each $i \in [n]$ he makes at most t_i queries to oracle $\mathcal{O}_i$.

Oracle programming. Let $f : \{0,1\}^* \to \{0,1\}^\sigma$ be a function. Let $\mu := \{(x_i, y_i)\}_i$ be a query-answer list, where $\forall i, (x_i, y_i) \in \{0,1\}^* \times \{0,1\}^\sigma$. Then we define the programming of f relative to query-answer list μ to be the function

$$f[\mu](x) := \begin{cases} y_i & \text{if } x = x_i \text{ for some } i, \\ f(x) & \text{otherwise.} \end{cases}$$

Relations. A relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is a set of tuples $(\mathbb{x}, \mathbb{w})$, where $\mathbb{x}$ is called the instance and $\mathbb{w}$ is called the witness. A relation $\mathcal{R}$ naturally induces a language $\mathcal{L}(R)$, which is the set of instances $\mathbb{x}$ such that there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$.

Cryptographic models. We use the word "plain" as an adjective to signify the absence of a common reference string (CRS) in a cryptographic model (*plain* standard model, *plain* ROM, etc.). We refer to the standard model as one that does not make use of any oracles and relies only on standard cryptographic assumptions.

2.2 The random oracle model

We provide the definition of the random oracle model, a framework for designing and analyzing cryptographic protocols introduced in [BR93].

Random oracles. For $\sigma \in \mathbb{N}$, we denote by $\mathcal{U}(\sigma)$ the uniform distribution over all functions of the form $f\colon \{0,1\}^* \to \{0,1\}^\sigma$. Equivalently, if f is sampled from $\mathcal{U}(\sigma)$, then for every input x it holds that $y := f(x)$ is a uniformly random σ-bit string (sampled independently for each input). We refer to f sampled from $\mathcal{U}(\sigma)$ as a *random oracle* with output size σ.

We denote sampling f from $\mathcal{U}(\sigma)$ as

$$f \leftarrow \mathcal{U}(\sigma) \ .$$

The random function f idealizes a real-world hash function because each input in $\{0,1\}^*$ is mapped to a random output, independently of the output for any other input.

Definition 2.1. *The* **ROM with output size** $\sigma \in \mathbb{N}$ *is the model where all parties (honest and malicious) are oracle algorithms and they are each given query access to the same function* $f\colon \{0,1\}^* \to \{0,1\}^\sigma$ *sampled from the distribution* $\mathcal{U}(\sigma)$*.*

Security analyses of cryptographic protocols in the ROM differ depending on which classes of adversaries they consider for establishing security. The setting of *bounded-query* adversaries is where an adversary trying to attack a cryptographic protocol is all-resourceful, with the only restriction being that he can query the random oracle a bounded number of times. The setting of *bounded-time* adversaries, which is weaker and implies the former notion, additionally restricts the running time of the adversary. This bounded-query model is called the "pure" ROM, where protocols can leverage, and only leverage, the same random function given to everyone.

We use the symbol $t \in \mathbb{N}$ to denote the adversary's query bound. Intuitively, as t grows, so does the ability of a t-query adversary to break the security of a given cryptographic protocol.

2.3 Non-interactive arguments in the ROM

A *non-interactive argument* (NARG) in the ROM is a tuple

$$\mathsf{NARG} = (\mathcal{P}, \mathcal{V}) \ ,$$

where $\mathcal{P}$ is an oracle algorithm known as the *argument prover* and $\mathcal{V}$ is an oracle algorithm known as the *argument verifier*.

A random oracle f is sampled from the distribution $\mathcal{U}(\sigma)$, for a given output size $\sigma \in \mathbb{N}$. Anyone, including the argument prover $\mathcal{P}$ and the argument verifier $\mathcal{V}$, can query f. The argument prover $\mathcal{P}$ receives an instance $\mathbb{x}$ and witness $\mathbb{w}$ as input and outputs an argument string π. The argument verifier $\mathcal{V}$ receives as input the instance $\mathbb{x}$ and argument string π and outputs a bit denoting whether to accept (the bit is 1) or reject (the bit is 0). See Figure 1 for a diagram.

We say that $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ is a non-interactive argument in the ROM for a relation $\mathcal{R}$ if it satisfies two main properties: *completeness* and *soundness*.

Below, we provide the formal definitions of the notions above.

Definition 2.2 (completeness)**.** *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is* **(perfectly) complete** *if for every output size* $\sigma \in \mathbb{N}$ *of the random oracle and instance-witness pair* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$*,*

$$\Pr\left[\mathcal{V}^f(\mathbb{x}, \pi) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\sigma) \\ \pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w}) \end{array}\right] = 1 \ .$$

The probability is taken over the sampling of f *and any randomness of the argument prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$*.*

Definition 2.3 (adaptive soundness)**.** *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **adaptive soundness error** ϵ_{ARG} *if for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, query bound* $t \in \mathbb{N}$*,* t*-query malicious argument prover* $\tilde{\mathcal{P}}$*, and instance size bound* $n \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leqslant n \\ \wedge\ \mathbb{x} \notin \mathcal{L}(\mathcal{R}) \\ \wedge\ \mathcal{V}^f(\mathbb{x}, \pi) = 1 \end{array} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\sigma) \\ (\mathbb{x}, \pi) \leftarrow \tilde{\mathcal{P}}^f \end{array}\right] \leqslant \epsilon_{\mathrm{ARG}}(\sigma, t, n) \ .$$

The probability is taken over the sampling of f *and any randomness of the argument verifier* $\mathcal{V}$*.*

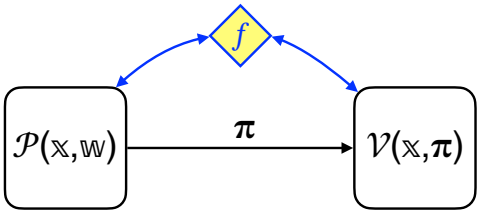


Figure 1: Diagram of a non-interactive argument in the random oracle model.

2.4 Knowledge soundness

While soundness (Definition 2.3) is necessary for foundational definitions, more is needed for useful applications. This section discusses these shortcomings and defines knowledge soundness in its two fundamental forms.

Knowledge soundness is a stronger security notion. The requirement states that if an argument prover convinces the argument verifier to accept then with overwhelming probability, the argument prover "knows" a valid witness for that instance. We refer to proofs/arguments satisfying this property as proofs/arguments of knowledge [GMR85; FS89; BG93].

Applications often require the use of this stronger notion. Indeed, the language associated with the NARG's relation could be total (i.e., $\mathcal{L}(\mathcal{R}) = \{0,1\}^*$). In this case, proving that an instance is in the language is much less involved than proving the knowledge of a certifying witness. In the case of the former, the trivial proof system is good enough: empty argument strings and an always-accepting verifier make up for a perfectly complete and perfectly sound proof system. This emphasizes the importance of knowledge soundness.

Knowledge soundness is formalized by requiring an efficient algorithm called the *extractor*, which finds a witness from the argument prover, thereby proving the prover's knowledge. We define two variations of knowledge soundness below: the more robust notion of straight-line knowledge

soundness and the weaker notion of rewinding knowledge soundness. In both cases, the extractor is given the query trace tr of the argument prover with the random oracle. The two notions then vary depending on the additional information the extractor obtains. Intuitively, the more resourceful the extractor is, the easier the extraction process becomes, and consequently, the easier it is for a proof system to satisfy this notion.

Straightline knowledge soundness. The definition below captures the stronger variation of knowledge soundness known as *straightline*. The strength of the notion comes from the fact that the extractor only needs information associated with a single execution of the argument prover. In more detail, the extractor finds the witness given only the argument prover's output and his query-answer trace with the random oracle.

Definition 2.4. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **straightline knowledge soundness error** κ_{ARG} *if there exists a polynomial-time deterministic algorithm* $\mathcal{E}$ *(the* extractor*) such that for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, query bound* $t \in \mathbb{N}$*, t-query deterministic argument prover* $\tilde{\mathcal{P}}$*, and instance size bound* $n \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leqslant n & f \leftarrow \mathcal{U}(\sigma) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & (\mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^f \\ \wedge\ \mathcal{V}^f(\mathbb{x}, \pi) = 1 & \mathbb{w} \leftarrow \mathcal{E}(\mathbb{x}, \pi, \mathsf{tr}) \end{array}\right] \leqslant \kappa_{\mathrm{ARG}}(\sigma, t, n) \ .$$

Rewinding knowledge soundness. The definition below formalizes a relaxation of Definition 2.4, where the extraction process receives as input an instance $\mathbb{x}$, an argument string π, the query-answer trace tr of the prover with the random oracle, and the prover as a black-box. The extractor is also granted access to the random oracle and can have access to randomness. Furthermore, the extractor's error is permitted to depend on the argument prover's failure probability in convincing the verifier. For succinctness, we omit the definition of this last notion.

Having black-box access to the prover allows the extractor to re-run the prover multiple times on inputs of its choice, using the random oracle to facilitate this. This makes extraction of the witness much easier. For instance, in some cases, having as little as two accepting argument strings enables finding a witness by solving a system of linear equations [Sch90].

Definition 2.5. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **rewinding knowledge soundness error** κ_{ARG} **with extraction time** $\mathsf{et}_{\mathrm{ARG}}$ *if there exists a probabilistic algorithm* $\mathcal{E}$ *(the* extractor*) such that for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, query bound* $t \in \mathbb{N}$*, t-query deterministic argument prover* $\tilde{\mathcal{P}}$ *with failure probability* $\delta_{\tilde{\mathcal{P}}}$ *and running time* $\tau_{\tilde{\mathcal{P}}}$*, and instance size bound* $n \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leqslant n & f \leftarrow \mathcal{U}(\sigma) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & (\mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \tilde{\mathcal{P}}^f \\ \wedge\ \mathcal{V}^f(\mathbb{x}, \pi) = 1 & \mathbb{w} \leftarrow \mathcal{E}^f(\mathbb{x}, \pi, \mathsf{tr}, \tilde{\mathcal{P}}) \end{array}\right] \leqslant \kappa_{\mathrm{ARG}}(\sigma, t, n, \delta_{\tilde{\mathcal{P}}}(\sigma, n)) \ .$$

Moreover, $\mathcal{E}$ *runs in expected time* $\mathsf{et}_{\mathrm{ARG}}(\sigma, t, n, \delta_{\tilde{\mathcal{P}}}(\sigma, n), \tau_{\tilde{\mathcal{P}}}(\sigma, n))$ *(over the given inputs and internal randomness).*

Definition 2.4 implies Definition 2.5 because in the latter, the extractor receives more inputs.

2.5 Zero-knowledge

In the constructions studied later in this work, it is necessary to establish security notions that protect the (honest) party providing the argument string from a malicious party wishing to learn information it is not entitled to know. The security notion that captures this is *zero-knowledge*, first formally introduced in [GMR85]. Below, we discuss zero-knowledge and provide necessary definitions for its formalization in the pure random oracle model.

In the context of NARGs, the instance $\mathbb{x}$ is an input to the argument prover and verifier, and the witness $\mathbb{w}$ is an input to the argument prover only. To ascertain the privacy of the honest prover, we aim for the following: π does not reveal any information about $\mathbb{w}$ beyond what is implied by the fact that $\mathbb{x}$ is in the language $\mathcal{L}(\mathcal{R})$.

To capture this requirement, it is necessary to show that there exists an efficient probabilistic algorithm $\mathcal{S}$, called the *simulator*, which outputs argument strings given *only* an instance, and that these *simulated* argument strings are indistinguishable from those generated by the (witness-knowing) prover. Informally, an adversary that can query a proving oracle (additionally to the random oracle) on a bounded number of instance-witness pairs of his choosing should be unable to tell (up to a small error) whether the argument strings he obtains in return come from the prover, which generates argument strings using both the instance *and* the witness, or the simulator, which generates argument strings based on the instance only.

To formalize the above, we define two experiments: a real-world experiment and a simulated-world experiment. Based on these, we will formally define zero-knowledge in the ROM.

Definition 2.6. *Let $\sigma \in \mathbb{N}$ be the output size of the random oracle, $n \in \mathbb{N}$ an instance size bound, and $\mathcal{A}$ an adversary. The* **real-world experiment** *is the distribution defined as follows:*

$\mathsf{ZKExpReal}(\sigma, n, \mathcal{A})$*:*

1. *Sample a random oracle $f \leftarrow \mathcal{U}(\sigma)$.*
2. *Emulate an execution of $\mathcal{A}$ and do the following to answer his queries.*
   (a) *random oracle query x: answer with $y := f(x)$.*
   (b) *proving oracle query $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with $|\mathbb{x}| \leqslant n$: answer with $\pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w})$.*
3. *Output $\mathcal{A}$'s output.*

Definition 2.7. *Let $\sigma \in \mathbb{N}$ be the output size of the random oracle, $n \in \mathbb{N}$ an instance size bound, $\mathcal{A}$ an adversary, and $\mathcal{S}$ a simulator. The* **simulated-world experiment** *is the distribution defined as follows:*

$\mathsf{ZKExpSim}(\sigma, n, \mathcal{A}, \mathcal{S})$*:*

1. *Sample a random oracle $f \leftarrow \mathcal{U}(\sigma)$.*
2. *Initialize an empty query-answer list μ_{all}.*
3. *Emulate an execution of $\mathcal{A}$ and do the following to answer his queries.*
   (a) *random oracle query x: answer with $y := f[\mu_{\text{all}}](x)$.*
   (b) *proving oracle query $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with $|\mathbb{x}| \leqslant n$: sample $(\pi, \mu) \leftarrow \mathcal{S}^{f[\mu_{\text{all}}]}(\mathbb{x})$, append μ to μ_{all} provided that μ and μ_{all} do not share queries, and answer with π. (Abort if μ and μ_{all} share queries.)*
4. *Output $\mathcal{A}$'s output.*

Before defining zero-knowledge, we quickly highlight the fundamental differences between the two experiments. Firstly, in the real-world experiment, the proving query is answered by the honest

prover, and in the simulated-world experiment, the proving query is answered by the simulator. Secondly, we allow the simulator to program the random oracle. This is called the *explicitly-programmable random oracle model* (EPROM) and is a necessary condition for achieving zero-knowledge, as without it, the property is too strong to satisfy (see [CY24, Chapter 6.6]). Intuitively, granting the simulator this capability is acceptable as long as the programmed queries appear random enough, ensuring the adversary cannot distinguish between a genuinely random query and a programmed query. As such, Definition 2.7 includes the additional components necessary for programming the random oracle.

Below, we formalize the definition of zero-knowledge and provide all the parameters upon which the security bound depends.

Definition 2.8. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **zero-knowledge error** z_{ARG} *(in the EPROM) if there exists a probabilistic polynomial-time simulator* $\mathcal{S}$ *such that, for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, random oracle query bound* $t \in \mathbb{N}$*, argument string query bound* $t_{\mathcal{P}} \in \mathbb{N}$*,* $(t, t_{\mathcal{P}})$*-query admissible adversary* $\mathcal{A}$*, and instance size bound* $n \in \mathbb{N}$*, the following two distributions are* $z_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n)$*-close in statistical distance*

$$\mathsf{ZKExpReal}(\sigma, n, \mathcal{A}) \quad \textit{and} \quad \mathsf{ZKExpSim}(\sigma, n, \mathcal{A}, \mathcal{S}) \ .$$

Note that an admissible adversary's queries to the random oracle can be anything, whereas queries to the proving oracle must be valid instance-witness pairs where the instance has bounded size.

Computational zero-knowledge. Definition 2.8 encompasses an adversary that is bounded only in his query budget to the random oracle and the proving oracle. For applications that often rely on hardness assumptions, it may suffice for the zero-knowledge property to hold for adversaries that are additionally computationally bounded.[1] Since we consider non-uniform adversaries, the meaningful metric to bound is the adversary's size.

Definition 2.9. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **computational zero-knowledge error** z_{ARG} *(in the EPROM) if everything is as in Definition 2.8, but the adversary's size is additionaly bounded by* $s \in \mathbb{N}$*. The statistical distance of the distributions,* $z_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n, s)$*, additionally depends on this size.*

In Definitions 2.8 and 2.9, the case of $t_{\mathcal{P}} = 1$ is referred to as *one-time (computational) zero-knowledge*. The case of $t_{\mathcal{P}} > 1$ strictly strengthens the notion and is referred to as *multi-instance (computational) zero-knowledge*. Intuitively, the more argument strings an adversary can see, the greater the chance for him to identify discrepancies between real and simulated argument strings. Therefore, for stronger security guarantees, it is desirable to have z_{ARG} increase slowly as $t_{\mathcal{P}}$ increases.

[1] We note that the computational bound implicitly bounds the total query budget of the adversary.

3 Simulation security

In this section, we discuss and define ways to further strengthen the security properties of NARGs in the random oracle model. We focus on simulation security, a term we use to refer to simulation soundness and simulation knowledge soundness collectively.

Simulation soundness and simulation knowledge soundness have emerged as concepts in cryptographic research in the context of *non-malleability* [DDN91; DDN00; Sah99]. Informally, non-malleability states that no man-in-the-middle should be capable of increasing his chances of proving a statement, even if he obtains an argument string for a related statement [JP11].

So far, soundness definitions have been relative to an adversary operating in isolation. This approach does not provide guarantees in scenarios where the adversary can observe argument strings for other related instances, potentially gaining an advantage. In many applications, such resources exist, blockchains being an example. Therefore, we must establish security guarantees for an adversary given access to such a resource. We underline the significance of this requirement with an example below, adapted from [Sah99].

Example 3.1 (Malleable NARG)**.** *Consider a perfectly complete, knowledge sound, zero-knowledge* $\mathsf{NARG} := (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$*. Consider a relation*

$$\mathcal{R}' := \left\{((\mathbb{x}_1, \mathbb{x}_2), (\mathbb{w}_1, \mathbb{w}_2)) \;\middle|\; (\mathbb{x}_1, \mathbb{w}_1) \in \mathcal{R} \wedge (\mathbb{x}_2, \mathbb{w}_2) \in \mathcal{R}\right\} .$$

We define $\mathsf{NARG}' := (\mathcal{P}', \mathcal{V}')$ *for* $\mathcal{R}'$*, where the prover and the verifier work as shown below.*

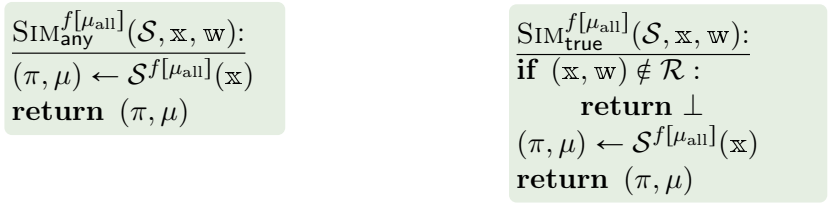
$\underline{\mathcal{P}'^{f}((\mathbb{x}_1, \mathbb{x}_2), (\mathbb{w}_1, \mathbb{w}_2)):}$
$\pi_1 \leftarrow \mathcal{P}^{f}(\mathbb{x}_1, \mathbb{w}_1)$
$\pi_2 \leftarrow \mathcal{P}^{f}(\mathbb{x}_2, \mathbb{w}_2)$
return (π_1, π_2)

$\underline{\mathcal{V}'^{f}((\mathbb{x}_1, \mathbb{x}_2), (\pi_1, \pi_2)):}$
return $\mathcal{V}^{f}(\mathbb{x}_1, \pi_1)$
$\wedge \mathcal{V}^{f}(\mathbb{x}_2, \pi_2)$

Trivially, NARG' *preserves all the security properties of* NARG*. However, the proof system is malleable, as we discuss now. Consider an adversary* A *that knows a witness* $\mathbb{w}$ *for instance* $\mathbb{x}$ *such that* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$*. Naturally,* A *can obtain an argument string* π *for the statement. Suppose* A *now observes an argument string* (π_1, π_2) *for* $(\mathbb{x}_1, \mathbb{x}_2) \in \mathcal{L}(\mathcal{R}')$*. Thanks to this, he is capable of providing an argument string* (π, π_2) *for* $(\mathbb{x}, \mathbb{x}_2)$*, without knowing* $\mathbb{w}_2$*, thereby proving something he was not capable of proving beforehand.*

Classical definitions of non-malleability are phrased as man-in-the-middle attacks, where an adversary acts as a mediator between a prover and a verifier: the adversary, acting as a verifier for the prover, uses his interaction with the prover to produce dishonest proofs for the verifier, for whom, in turn, he acts as a prover. These definitions are only sometimes helpful when the proof system satisfies the zero-knowledge property. In particular, they do not necessarily involve the zero-knowledge simulator for the given proof system.

Having the ability to encapsulate the zero-knowledge simulator's behavior is, however, compelling, as it allows for proving strong security properties. Let us consider a cryptographic construction that uses a zero-knowledge NARG as a building block. In a security analysis of such a construction, when one invokes the zero-knowledge property, the adversary's view changes from one with real argument strings to one with simulated argument strings. As such, the soundness and knowledge soundness properties should continue to hold if the adversary has access to simulated



(a) (b)

Figure 3: Implementations for the different flavors of the simulation oracle.

argument strings, appropriately limiting his advantage in the simulated scenario. We now discuss two fundamental ways this differs from a real scenario.

Firstly, the simulator has the "superpower" to program the random oracle. The adversary should not be able to leverage the oracle's programming to prove things he should be incapable of. In more detail, argument strings, despite being simulated, should not help the adversary in creating valid argument strings for false statements (in the case of soundness), or it should not help him with creating valid argument strings for instances for which he does not know a witness (in the case of knowledge soundness).

Secondly, it is possible to invoke the simulator on any instance, including one that is not in the language. The zero-knowledge property gives no guarantees on the distribution of an argument string produced by the zero-knowledge simulator invoked on instances not in the language. Furthermore, as shown in [CY24, Chapter 6.7], for hard languages, the simulator produces convincing argument strings *even for false statements*. Such invokations be of no use to the adversary.

Informally, for simulation security, we grant the adversary access to a simulation oracle. He may query this oracle a bounded number of times, allowing him to see simulated argument strings π' for instances $\mathbb{x}'$ of his choosing before he outputs $\mathbb{x}$ and π such that $\mathbb{x}$ is not among one of the instances previously queried. Since we (necessarily) allow the zero-knowledge simulator to (explicitly) program the random oracle (see Section 2.5), the definition of simulation security considers simulators that (explicitly) program the random oracle.

We consider two flavors of simulation security, as first proposed by [DHLW10], which differ in the predicate dictating the type of queries for which the adversary can get simulated argument strings. In both cases, the adversary's queries to the simulation oracle must be instances $\mathbb{x}$ accompanied by a witness $\mathbb{w}$. We depict them in Figure 3 and describe these below.

- *Any* simulation security (Figure 3a): the simulation oracle unconditionally returns the simulated argument string. In particular, the adversary can see simulated argument strings for false statements.
- *True* simulation security (Figure 3b): the simulation oracle returns the simulated argument string only if the instance-witness pair is in the relation. Otherwise, it fails.

In practice, designing NARGs satisfying *any* simulation security can be more challenging than designing NARGs satisfying *true* simulation security. [DHLW10] show that a *true* simulation knowledge sound zero-knowledge NARG can be constructed from a labeled CCA-secure encryption scheme and a standard zero-knowledge NARG. By considering these variants, we show that NARGs satisfying the weaker property can nevertheless be valuable for cryptographic constructions, as shown

in Sections 4 and 5.

We first define the experiment in which the adversary participates, based on which we define the notion of simulation knowledge soundness. Then, we discuss variations to the definitions.

Definition 3.2. *Let $\sigma \in \mathbb{N}$ be the output size of the random oracle, $n \in \mathbb{N}$ an instance size bound, $\mathcal{A}$ an adversary, $\mathcal{S}$ a simulator, and* $\mathsf{FLAVOR} \in \{\mathsf{any}, \mathsf{true}\}$*. The* **FLAVOR-simulation security experiment** *is the distribution defined as follows:*

$\mathsf{SSExp}_{\mathsf{FLAVOR}}(\sigma, n, \mathcal{A}, \mathcal{S})$*:*

1. *Sample a random oracle $f \leftarrow \mathcal{U}(\sigma)$.*
2. *Initialize an empty query-answer list μ_{all}.*
3. *Emulate an execution of $\mathcal{A}$ and do the following to answer his queries.*
   (a) *Random oracle query x: answer with $y := f[\mu_{\text{all}}](x)$.*
   (b) *Simulation oracle query $(\mathbb{x}, \mathbb{w})$, with $|\mathbb{x}| \leqslant n$: sample $(\pi, \mu) \leftarrow \mathit{SIM}^{f[\mu_{\text{all}}]}_{\mathsf{FLAVOR}}(\mathcal{S}, \mathbb{x}, \mathbb{w})$, append μ to μ_{all} provided that μ and μ_{all} do not share queries, and answer with π. (Abort if μ and μ_{all} share queries.)*
4. *Output $\mathcal{A}$'s output.*

Definition 3.3 (Straightline simulation knowledge soundness)**.** *For* $\mathsf{FLAVOR} \in \{\mathsf{any}, \mathsf{true}\}$*, a non-interactive argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ has* **straightline FLAVOR-simulation knowledge soundness error** $\kappa^{\text{SIM}}_{\text{ARG}}$ *(in the EPROM) relative to the probabilistic polynomial-time simulator $\mathcal{S}$ if there exists a polynomial-time deterministic extractor algorithm $\mathcal{E}$ such that for every output size $\sigma \in \mathbb{N}$ of the random oracle, random oracle query bound $t \in \mathbb{N}$, argument string query bound $t_{\mathcal{P}} \in \mathbb{N}$, $(t, t_{\mathcal{P}})$-query admissible adversary $\mathcal{A}$, and instance size bound $n \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leqslant n & (\mathbb{x}, \pi) \xleftarrow{f, \mu_{\text{all}}, \mathsf{tr}_{\text{RO}}} \mathsf{SSExp}_{\mathsf{FLAVOR}}(\sigma, n, \mathcal{A}, \mathcal{S}) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathbb{w} \leftarrow \mathcal{E}(\mathbb{x}, \pi, \mathsf{tr}_{\text{RO}}) \\ \wedge\ \mathcal{V}^{f[\mu_{\text{all}}]}(\mathbb{x}, \pi) = 1 & \end{array}\right] \leqslant \kappa^{\text{SIM}}_{\text{ARG}}(\sigma, t, t_{\mathcal{P}}, n) \ ,$$

where f is the random oracle sampled in $\mathsf{SSExp}_{\mathsf{FLAVOR}}$, μ_{all} is the list containing the programming of the random oracle by $\mathcal{S}$, and tr_{RO} is the query-answer trace of $\mathcal{A}$ with the random oracle. Above, $\mathcal{A}$ is admissible if every instance queried to the simulation oracle has size at most n and always outputs an instance $\mathbb{x}$ not previously queried to the simulation oracle.

Below, we discuss the different "tweaks" to this definition and how these affect the strength of the security notion.

Remark 3.4. We could consider a variant of the above definition where the extractor additionally obtains the trace tr_{PR} of queries made by the adversary to the proving oracle. However, this is beyond the scope of this work.

Rewinding simulation knowledge soundness. Similarly to Definition 2.5, we may consider the relaxed definition of rewinding simulation knowledge soundness, where the extractor is randomized, receives access to the random oracle, receives the adversary as a black-box, and runs in *expected* polynomial time. The experiment from Definition 3.3 in this case becomes

$$\left[\begin{array}{l} (\mathbb{x}, \pi) \xleftarrow{f, \mu_{\text{all}}, \mathsf{tr}_{\text{RO}}} \mathsf{SSExp}_{\mathsf{FLAVOR}}(\sigma, n, \mathcal{A}, \mathcal{S}) \\ \mathbb{w} \leftarrow \mathcal{E}^{f}(\mathbb{x}, \pi, \mathsf{tr}_{\text{RO}}, \mathcal{A}) \end{array}\right] .$$

The error bound also depends on the adversary's failure probability $\delta_{\mathcal{A}}(\sigma, n)$. We define this below.

Definition 3.5 (Rewinding simulation knowledge soundness)**.** *For* $\mathsf{FLAVOR} \in \{\mathsf{any}, \mathsf{true}\}$*, a non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **rewinding** FLAVOR**-simulation knowledge soundness error** $\kappa^{\mathrm{SIM}}_{\mathrm{ARG}}$ *(in the EPROM) relative to the probabilistic polynomial-time simulator* $\mathcal{S}$ *if given the modifications stated above, the probability from Definition 3.3 is at most* $\kappa^{\mathrm{SIM}}_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n, \delta_{\mathcal{A}}(\sigma, n))$*.*

Simulation soundness. We define below the notion of simulation soundness. Intuitively, the definition states that an adversary taking part in the experiment from Definition 3.2 should have bounded success probability in an attempt to provide a valid argument string for a false statement. The probability outcome in the probability statement is the same as in Definition 2.3, except that the verifier has access to the programmed random oracle. We may think of the following definition as a case of Definition 3.3 with an all powerful extractor.

Definition 3.6 (Simulation soundness)**.** *For* $\mathsf{FLAVOR} \in \{\mathsf{any}, \mathsf{true}\}$*, a non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* FLAVOR**-simulation soundness error** $\epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}$ *(in the EPROM) relative to the probabilistic polynomial-time simulator* $\mathcal{S}$ *if for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, random oracle query bound* $t \in \mathbb{N}$*, argument string query bound* $t_{\mathcal{P}} \in \mathbb{N}$*,* $(t, t_{\mathcal{P}})$*-query admissible adversary* $\mathcal{A}$*, and instance size bound* $n \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leqslant n & \\ \wedge\ \mathbb{x} \notin \mathcal{L}(\mathcal{R}) & (\mathbb{x}, \pi) \xleftarrow{f, \mu_{\mathrm{all}}, \mathsf{tr}_{\mathrm{RO}}} \mathsf{SSExp}_{\mathsf{FLAVOR}}(\sigma, n, \mathcal{A}, \mathcal{S}) \\ \wedge\ \mathcal{V}^{f[\mu_{\mathrm{all}}]}(\mathbb{x}, \pi) = 1 & \end{array}\right] \leqslant \epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n) \ ,$$

where f *is the random oracle sampled in* $\mathsf{SSExp}_{\mathsf{FLAVOR}}$*,* μ_{all} *is the list containing the programming of the random oracle by* $\mathcal{S}$*, and* $\mathsf{tr}_{\mathrm{RO}}$ *is the query-answer trace of* $\mathcal{A}$ *with the random oracle. Above,* $\mathcal{A}$ *is admissible if every instance queried to the simulation oracle has size at most* n *and always outputs an instance* $\mathbb{x}$ *not previously queried to the simulation oracle.*

Computational simulation security. For Definitions 3.3, 3.5 and 3.6, the error needs to hold with respect to adversaries bounded only in their query budget to the oracles. Just as in Definition 2.9, we provide a variant, where additionally the adversary is bounded in computational resources.

Definition 3.7. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **computational simulation security** *if each of the error bounds from Definitions 3.3, 3.5 and 3.6 holds for an adversary of size at most* $s \in \mathbb{N}$*. The error bounds,* $\kappa^{\mathrm{SIM}}_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n, s), \kappa^{\mathrm{SIM}}_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n, \delta_{\mathcal{A}}(\sigma, n), s)$*, and* $\epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}(\sigma, t, t_{\mathcal{P}}, n, s)$*, respectively, additionally depend on* s*.*

4 Signature schemes

A signature scheme is a cryptographic primitive that enables a party to use a secret key to sign messages so that anyone else can verify their authenticity via a corresponding public key. Signatures aim to be unforgeable, which means that no efficient attacker who knows only the public key can produce valid signatures for new messages (i.e., messages that have not been previously signed using the secret key).

In this section, we formally define a signature scheme in the ROM and state the security properties that such a scheme should satisfy. We then construct a signature scheme from two ingredients: (1) hard relations (e.g., one-way functions); and (2) non-interactive arguments in the ROM. The non-interactive argument must satisfy computational zero-knowledge and computational *true*-simulation knowledge soundness. This application shows how to protect against an adversary with access to a simulation oracle that he can only query on true instances. The signature scheme satisfies existential unforgeability under chosen message attacks (EUF-CMA) [GMR88].

Katz and Vaikuntanathan [KV09] first introduced the framework used in this construction. The approach was subsequently refined and generalized by Dodis et al. [DHLW10]. It was adapted to the random oracle model in [FKMV12]. These works studied security in the context of "memory attacks" [AGV09], ensuring resilience in the presence of bounded leakage of the secret key. In this work, however, we demonstrate the EUF-CMA security of the construction without addressing the notion of key-leakage, which we leave for future work. Instead, we focus on the concrete security parameters of the security notions we analyze.

The section is structured as follows.

- In Section 4.1, we define the notion of a signature scheme in the ROM.
- In Section 4.2, we define hard-relations: a building block for our construction.
- In Section 4.3, we describe the construction of the signature scheme in the ROM.
- In Section 4.4, we analyze the security of the construction.
- In Section 4.5, we discuss lower bounds related to the construction.

4.1 Definition

A *signature scheme* in the ROM is a tuple of algorithms

$$\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$$

and has two parameters: a security parameter $\lambda \in \mathbb{N}$ and a message length $\ell \in \mathbb{N}$. We use the notation $\mathsf{SIG}[\lambda, \ell]$ to make these parameters explicit. A random oracle f is sampled from the distribution $\mathcal{U}(\lambda)$, and the algorithms work as follows.

- The *key generator algorithm* $\mathsf{SIG.Gen}$, given as input the security parameter λ (represented in unary), outputs a public key pk and a secret key sk.
- The *signing algorithm* $\mathsf{SIG.Sign}$, given query access to f and input the secret key sk and a message $m \in \{0,1\}^\ell$, outputs a signature σ.
- The *verification algorithm* $\mathsf{SIG.Verify}$, given query access to f and input the public key pk, message $m \in \{0,1\}^\ell$, and signature σ, outputs 1 if and only if the signature is valid.

A signature scheme must satisfy two properties. *Completeness* states that signatures produced by the signing algorithm make the verification algorithm accept. *Unforgeability* states that no

efficient attacker can forge signatures. The notion of unforgeability comes in several flavors. In this work, we focus on existential unforgeability under chosen message attacks (EUF-CMA). The definition considers a computationally bounded attacker with access to a public key and a "signing oracle", which it can query on adaptively chosen messages to obtain signatures created with the corresponding secret key. EUF-CMA states that such an attacker should be unable to produce a valid signature for a new message (one that has not been previously queried to the signing oracle).

Definition 4.1. $\mathsf{SIG} := \mathsf{SIG}[\lambda, \ell]$ *is* **(perfectly) complete** *if, for every message* $m \in \{0,1\}^\ell$*,*

$$\Pr\left[\mathsf{SIG.Verify}^f(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SIG.Gen}(1^\lambda) \\ \sigma \leftarrow \mathsf{SIG.Sign}^f(\mathsf{sk}, m) \end{array}\right] = 1 \ .$$

Definition 4.2. $\mathsf{SIG} := \mathsf{SIG}[\lambda, \ell]$ *has* **unforgeability error** ϵ_{SIG} *if for every query bound* $t \in \mathbb{N}$ *for the random oracle, query bound* $t_{\mathrm{SIG}} \in \mathbb{N}$ *for the signing oracle, adversary size bound* $s \in \mathbb{N}$*, and* (t, t_{SIG})*-query admissible algorithm* A *of size* s*,*

$$\Pr\left[\mathsf{SIG.Verify}^f(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SIG.Gen}(1^\lambda) \\ (m, \sigma) \leftarrow A^{f, \mathsf{SIG.Sign}^f(\mathsf{sk}, \cdot)}(\mathsf{pk}) \end{array}\right] \leqslant \epsilon_{\mathrm{SIG}}(\lambda, \ell, t, t_{\mathrm{SIG}}, s) \ .$$

Above, A *is admissible if he always outputs a message* m *that was not queried to* $\mathsf{SIG.Sign}^f(\mathsf{sk}, \cdot)$*.*

Remark 4.3. In the above definition, the key generator algorithm $\mathsf{SIG.Gen}$ does not have query access to the random oracle f because the construction we consider does not need it. In general (for other constructions), it may be helpful for $\mathsf{SIG.Gen}$ to have query access to f.

4.2 Hard relations

In this section, we define a notion of a hard relation, an adaptation of the definition from [DHLW10, Definition 2.1], in the sense that we do not consider leakage resilience. Informally, a *hard relation* is a relation that has an instance-witness sampler with a small hardness error (defined below). We use hard relations to construct a signature scheme in Section 4.3. One-way functions naturally lead to hard relations, as explained in Remark 4.6.

Definition 4.4. *A probabilistic algorithm* G *is an* **instance-witness sampler** *for a relation* $\mathcal{R}$ *if, for every instance size bound* $n \in \mathbb{N}$*,* $G(1^n)$ *always outputs an instance-witness pair* $(\mathbb{x}, \mathbb{w})$ *in the relation* $\mathcal{R}$ *with* $|\mathbb{x}| \leqslant n$*.*

Definition 4.5. *An instance-witness sampler* G *for a relation* $\mathcal{R}$ *has* **hardness error** $\epsilon_{\mathcal{R}}$ *if for every instance size bound* $n \in \mathbb{N}$*, adversary size bound* $s \in \mathbb{N}$*, and every algorithm* A *of size* $s \in \mathbb{N}$*,*

$$\Pr\left[(\mathbb{x}, \mathbb{w}') \in \mathcal{R} \;\middle|\; \begin{array}{l} (\mathbb{x}, \mathbb{w}) \leftarrow G(1^n) \\ \mathbb{w}' \leftarrow A(\mathbb{x}) \end{array}\right] \leqslant \epsilon_{\mathcal{R}}(n, s) \ .$$

Remark 4.6 (Hard relations from one-wayness)**.** A *one-way function* [DH76] is a function for which images of random inputs are hard to invert. The existence of such a function is conjectured, as it would imply that $\mathsf{P} \neq \mathsf{NP}$.

Formally, a one-way function with hardness error ϵ is an efficient deterministic algorithm F such that for every security parameter $\lambda \in \mathbb{N}$, adversary size bound $s \in \mathbb{N}$, and algorithm A of size s,

$$\Pr\left[F(x') = y \;\middle|\; \begin{array}{l} x \leftarrow \{0,1\}^\lambda \\ y := F(x) \\ x' \leftarrow A(\lambda, y) \end{array} \right] \leqslant \epsilon(\lambda, s) \ .$$

A one-way function naturally leads to a corresponding instance-witness sampler G for the relation

$$\mathcal{R} := \left\{ \big((\lambda, y), x\big) : y = F(x) \right\} \ .$$

The instance-witness sampler $G(1^\lambda)$ samples $x \leftarrow \{0,1\}^\lambda$, computes $y := F(x)$, and outputs the instance $\mathbb{x} := (\lambda, y)$ and witness $\mathbb{w} := x$. If the one-way function has hardness error $\epsilon(\lambda, s)$, and its image set is $\{0,1\}^{\lambda'}$, then the instance witness sampler G for the relation $\mathcal{R}$ described here has hardness error $\epsilon_{\mathcal{R}}(n := \lambda + \lambda', s)$.

4.3 Construction

We now describe the construction of the signature scheme in the ROM. Let $\mathcal{R}$ relation and $\ell \in \mathbb{N}$ be a message length. We define the relation

$$\mathcal{R}_\ell := \left\{ \big((\mathbb{x}, m), \mathbb{w}\big) : (\mathbb{x}, \mathbb{w}) \in \mathcal{R} \wedge m \in \{0,1\}^\ell \right\} \ . \tag{1}$$

Consider the following two ingredients: (i) an instance-witness sampler G for a relation $\mathcal{R}$; and (ii) a non-interactive argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ for the associated relation $\mathcal{R}_\ell$.

Informally, the construction of the signature scheme $\mathsf{SIG} := \mathsf{SIG}[\lambda, \ell]$ is straightforward. A secret key sk is an instance-witness pair $(\mathbb{x}, \mathbb{w})$ in the relation $\mathcal{R}$ sampled using G, while the corresponding public key pk is the instance $\mathbb{x}$. Signing a message m consists of using the argument prover $\mathcal{P}$ to prove the statement "I know a secret $\mathbb{w}$ such that $\big((\mathbb{x}, m), \mathbb{w}\big) \in \mathcal{R}_\ell$" resulting in an argument string π, which is used as the signature ($\sigma := \pi$). Verifying a signature σ for a message m consists of using the argument verifier $\mathcal{V}$ to check that σ is a valid argument string for the instance $(\mathbb{x}, m)$.

Formally, let $\lambda \in \mathbb{N}$ be a security parameter and $f \in \mathcal{U}(\lambda)$ be a random oracle. We describe each of the algorithms of SIG below.

- $\mathsf{SIG.Gen}(1^\lambda)$:
  1. Sample an instance-witness pair $(\mathbb{x}, \mathbb{w}) \leftarrow G(1^\lambda)$ for the relation $\mathcal{R}$.
  2. Set the public key $\mathsf{pk} := \mathbb{x}$.
  3. Set the secret key $\mathsf{sk} := (\mathbb{x}, \mathbb{w})$.
  4. Output the key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{SIG.Sign}^f\big(\mathsf{sk} = (\mathbb{x}, \mathbb{w}), m \in \{0,1\}^\ell\big)$:
  1. Set the new instance $\mathbb{x}' := (\mathbb{x}, m)$.
  2. Sample the argument string $\pi \leftarrow \mathcal{P}^f(\mathbb{x}', \mathbb{w})$.
  3. Output the signature $\sigma := \pi$.
- $\mathsf{SIG.Verify}^f\big(\mathsf{pk} = \mathbb{x}, m \in \{0,1\}^\ell, \sigma = \pi\big)$:
  1. Set the new instance $\mathbb{x}' := (\mathbb{x}, m)$.
  2. Output $\mathcal{V}^f(\mathbb{x}', \pi)$.

4.4 Security analysis

The security of the signature scheme SIG constructed in Section 4.3 depends on several properties.

- The hardness of the relation $\mathcal{R}$. Intuitively, if it were easy to find a valid witness for an instance of $\mathcal{L}(\mathcal{R})$ (and thus for $\mathcal{L}(\mathcal{R}_\ell)$), then it would be easy to find a secret key for a given public key of SIG (which would allow forging signatures for SIG).

- The zero-knowledge property of NARG. Signatures in SIG are argument strings that attest to statements of the form "there exists a witness $\mathbb{w}$ such that for a given instance $\mathbb{x}$ and message m it holds that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$". Hence, if NARG were not zero-knowledge, then the argument string may, in principle, reveal $\mathbb{w}$, which allows deducing the secret key (and forging signatures).

- The simulation knowledge soundness property of NARG.

  The fact that NARG must have knowledge soundness is clear: the language $\mathcal{L}(\mathcal{R})$ (and thus $\mathcal{L}(\mathcal{R}_\ell)$) may contain all strings, in which case if NARG were merely sound (rather than knowledge sound) then argument strings in NARG could be empty. Anyone would be able to forge signatures in SIG. (The language $\mathcal{L}(\mathcal{R})$ may be trivial even when $\mathcal{R}$ arises from a one-way function as in Remark 4.6, which shows that this case is not contrived.)

  The requirement for the NARG to be *simulation* knowledge sound is less apparent. In the unforgeability experiment from Definition 4.2, the adversary gets access to a public key pair and an oracle that generates signatures using the secret key (unknown to the adversary). This additional information could help the adversary in producing a forgery. For example, if the argument strings were malleable, the adversary could modify an argument string obtained from the proving oracle into another one for a distinct message (which validates successfully under the same public key). Examples of NARG that are knowledge sound yet malleable are easy to construct (see Example 3.1). We use the stronger notion of simulation knowledge soundness to avoid this.

The following theorem formally captures the above intuition. The unforgeability error is upper bounded by a sum of errors corresponding to the above properties.

Theorem 4.7. *Consider the following two ingredients:*

- *G is an instance-witness sampler for a relation $\mathcal{R}$ with hardness error $\epsilon_{\mathcal{R}}$; and*
- *$\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ is a non-interactive argument for the relation $\mathcal{R}_\ell$ (see Equation 1) with computational* true*-simulation knowledge soundness error $\kappa^{\mathrm{SIM}}_{\mathrm{ARG}}$ and computational zero-knowledge error z_{ARG}.*

Then $\mathsf{SIG} := \mathsf{SIG}[\lambda, \ell]$ constructed in Section 4.3 is a signature scheme that (is perfectly complete and) has unforgeability error ϵ_{SIG} such that

$$\begin{aligned}\epsilon_{\mathrm{SIG}}(\lambda, \ell, t, t_{\mathrm{SIG}}, s) \leqslant & \epsilon_{\mathcal{R}}(\lambda + \ell, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{SIG}})) \\ & + \kappa^{\mathrm{SIM}}_{\mathrm{ARG}}(\lambda, t, t_{\mathrm{SIG}}, \lambda + \ell, s + \mathsf{poly}(\lambda, \ell)) \\ & + z_{\mathrm{ARG}}(\lambda, t, t_{\mathrm{SIG}}, \lambda + \ell, s + \mathsf{poly}(\lambda, \ell)) \ .\end{aligned}$$

We refer the reader to Appendix A for the proof of Theorem 4.7.

4.5 Lower bounds

Characterizing related lower bounds may also be helpful in this construction. Informally, for a given ingredient and its corresponding security property, we should not only prove that a given property contributes to the security of the construction but also show that it is necessary for its security. In particular, we wish to show that our properties are optimal, as using too strong properties could entail unnecessary overhead in practice.

Below, we demonstrate a strict lower bound on the hardness of the relation and discuss intuitions for lower bounds for the security properties of the underlying NARG.

Lower bound on the hardness of relation. Intuitively, it is necessary to use hard relations because if it were easy to find valid witnesses for instances of $\mathcal{L}(\mathcal{R})$, then so would be the forging of signatures. We formalize this below.

Suppose that for any $n, s \in \mathbb{N}$ there exists an adversary $A_{\mathcal{R}}$ of size s that guesses a valid witness for an instance of size n (wins the experiment in Definition 4.5) with probability *at least* $\epsilon_{\mathcal{R}}(n, s)$. We define an attacker A against the unforgeability property of SIG. The attacker receives a public key $\mathsf{pk} := \mathbb{x}$ sampled as $(\mathbb{x}, \mathbb{w}) \leftarrow G(1^\lambda)$, for which he attempts to forge a signature. He also receives query access to the random oracle and a signing oracle. He works as follows:

$\underline{A^{f,\mathsf{SIG.Sign}(\mathsf{sk},\cdot)}(\mathsf{pk} := \mathbb{x}):}$

1. Emulate the execution of $A_{\mathcal{R}}(\mathbb{x})$, to obtain a witness $\mathbb{w}'$.
2. Choose an arbitrary message $m \in \{0,1\}^\ell$ for which to forge a signature.
3. Run $\pi \leftarrow \mathcal{P}^f\big((\mathbb{x}, m), \mathbb{w}'\big)$, relaying random oracle queries and responses.
4. Return $(m, \sigma := \pi)$.

The attacker, A, has size at most $s + \mathsf{poly}(n, \ell)$, where the overhead comes from running the argument prover, as well as the computation of the argument string. He makes no use of the signing oracle and makes the same amount of queries to the random oracle as the honest prover $\mathcal{P}$. Furthermore, the instance passed to $A_{\mathcal{R}}$ is sampled exactly as in Definition 4.5, provided that A takes part in the unforgeability experiment of Definition 4.2 (the sampling of the instance and the public key is identical). It follows that the probability that $A_{\mathcal{R}}$ outputs a valid witness $\mathbb{w}'$ is at least $\epsilon_{\mathcal{R}}(n, s)$. Hence, the probability that A outputs a (m, σ) such that $\mathcal{V}^f((\mathbb{x}, m), \sigma) = 1$ is at least $\epsilon_{\mathcal{R}}(n, s)$. It follows that the unforgeability of SIG is lower bounded by the hardness of the relation it uses.

Relaxation of computational zero-knowledge. The underlying NARG in the construction has the zero-knowledge property. A weaker notion implied by the zero-knowledge property is witness indistinguishability, first introduced in [FS90]. Informally, witness indistinguishability states that if a statement has several witnesses, an adversary cannot distinguish which witness a prover uses to construct an argument string. We formally define this notion below in the random oracle model.

Definition 4.8. *A non-interactive argument* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has* **adaptive witness indistinguishability error** z_{ARG} *if for every output size* $\sigma \in \mathbb{N}$ *of the random oracle, query bound* $t \in \mathbb{N}$*, t-query admissible adversary* $\mathcal{A}$*, and instance size bound* $n \in \mathbb{N}$*, the random variables* X_0 *and* X_1 *are* $z_{\mathrm{ARG}}(\sigma, t, n)$*-close in statistical distance where*

$$X_b := \left\{ \mathsf{out} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\sigma) \\ (\mathbb{x}, \mathbb{w}_0, \mathbb{w}_1, \mathsf{aux}) \leftarrow \mathcal{A}^f \\ \pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w}_b) \\ \mathsf{out} \leftarrow \mathcal{A}(\mathsf{aux}, \pi) \end{array} \right\} .$$

Above, $\mathcal{A}$ is admissible if he always outputs $\mathbb{x}, \mathbb{w}_0, \mathbb{w}_1$ such that $(\mathbb{x}, \mathbb{w}_0), (\mathbb{x}, \mathbb{w}_1) \in \mathcal{R}$ and $|\mathbb{x}| \leqslant n$.

To show that witness indistinguishability is a strictly weaker notion, consider a relation where each instance has a *unique* witness.[2] For such a relation, a NARG where the prover simply reveals the witness is trivially witness indistinguishable, but not zero-knowledge. As such, in the construction from Section 4.3, it is generally insufficient to have the underlying NARG be witness indistinguishable because, depending on the choice of the one-way function, the underlying relation may be one with unique witnesses. Therefore, property that is strictly stronger than witness indistinguishability is required for this application.

For future work, we will leave the question of whether further relaxations to the zero-knowledge property are sufficient for this application.

Relaxation of computational *true*-simulation knowledge soundness. We do not show a tight lower bound on simulation soundness. However, we provide an intuition as to why it may be too strong of a property for the application at hand.

We notice that queries to the simulation oracle are instances with a particular structure throughout the security analysis. The simulator is queried exclusively on instance-witness pairs of the form $\big((\mathbb{x}, \cdot), \mathbb{w}\big)$, where $(\mathbb{x}, \mathbb{w})$ are fixed before the execution of A.

One approach to prove the lower bound is through a security reduction that uses a good adversary A_{SIM} against simulation knowledge soundness, appropriately simulates its environment, and uses its output to forge signatures. There are two obstacles to this approach: (1) A_{SIM} can forge signatures based on the programmed random oracle $f[\mu]$, which does not imply that the signatures would be valid relative to the non-programmed random oracle; and (2) A_{SIM} can create argument strings for any instance of $\mathcal{R}_\ell$ and in particular $\mathcal{R}$. In contrast, the adversary against EUF-CMA is bound to forge signatures for a $\mathbb{x} \in \mathcal{L}(\mathcal{R})$ it gets as input.

This latter suggests that it could be sufficient for NARG to be simulation sound with respect to only a subset of $\mathcal{R}_\ell$ (dictated by the instance $\mathbb{x}$), rather than the entire relation, and as such indicates that the property used in this construction might be too strong.

However, the practical significance of this property remains an open question. We are unaware of any examples where constructing a NARG with this restricted form of simulation soundness offers tangible efficiency advantages.

[2] A family of random permutations leads to such a relation.

5 Encryption schemes

An encryption scheme is a cryptographic primitive that enables a party to use a public key to encrypt messages so that only the holder of a corresponding secret key can decrypt and read them. Encryptions are secure, meaning no efficient attacker who knows only the public key can gain any meaningful information about the original message from its encrypted form.

Goldwasser and Micali [GM84] were the first to formalize security notions for encryption schemes, introducing the concept of semantic security. Informally, semantic security states that whatever can be efficiently computable about a plaintext, given its ciphertext, is also efficiently computable without it. The two most common notions that have evolved from this are security under chosen-plaintext attacks (CPA) and security under chosen-ciphertext attacks (CCA). The former notion is polynomially equivalent to semantic security. The latter is stronger and guarantees security in scenarios where an adversary can obtain decryptions to messages of his choice.

In this section, we formally define the notion of an encryption scheme in the ROM and define the corresponding notions of CPA and CCA security. We define these concepts using games played between a challenger and an adversary. We then construct an encryption scheme following the Naor-Young framework of double encryption [NY90], using a NARG satisfying computational *any*-simulation soundness and computational zero-knowledge, as well as a CPA secure encryption scheme. We then provide a concrete security reduction demonstrating the security of the scheme.

The section is structured as follows.

- In Section 5.1, we define the notion of an encryption scheme in the ROM.
- In Section 5.2, we describe the construction of the encryption scheme in the ROM.
- In Section 5.3, we analyze the security of the construction.
- In Section 5.4, we discuss lower bounds related to the construction.

5.1 Definition

An *encryption scheme* in the ROM is a tuple of algorithms

$$\mathsf{ENC} = (\mathsf{ENC.Gen}, \mathsf{ENC.Enc}, \mathsf{ENC.Dec})$$

and has two parameters: a security parameter $\lambda \in \mathbb{N}$, a message length $\ell \in \mathbb{N}$, and ciphertext length ℓ_c. We use the notation $\mathsf{ENC}[\lambda, \ell, \ell_c]$ when we wish to make these parameters explicit. A random oracle f is sampled from the distribution $\mathcal{U}(\lambda)$, and the algorithms work as follows.

- The *key generator algorithm* $\mathsf{ENC.Gen}$, given as input the security parameter λ (represented in unary), outputs a public key pk and a secret key sk each of length $\ell_{\mathsf{key}}(\lambda)$.
- The *encryption algorithm* $\mathsf{ENC.Enc}^f$, given query access to f and input the public key pk and a message $m \in \{0,1\}^\ell$, outputs a ciphertext $c \in \{0,1\}^{\ell_c}$.
- The *decryption algorithm* $\mathsf{ENC.Dec}^f$, given query access to f and input the secret key sk and ciphertext c, outputs a message $m \in \{0,1\}^\ell$ (or aborts if there is an error).

An encryption scheme must satisfy two properties. *Completeness* states that any message encrypted using the encryption algorithm and some public key, will be decrypted to itself, when using the corresponding secret key with the decryption algorithm. We provide the formal definition below:

Definition 5.1. $\mathsf{ENC} := \mathsf{ENC}[\lambda, \ell, \ell_c]$ *is* **(perfectly) complete** *if for every message* $m \in \{0,1\}^\ell$,

$$\Pr\left[\mathsf{ENC.Dec}^f(\mathsf{sk}, c) = m \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ c \leftarrow \mathsf{ENC.Enc}^f(\mathsf{pk}, m) \end{array}\right] = 1 \ .$$

Hiding, which is the other property we require, states that no efficient attacker can learn any information from encrypted messages. Formalizing this latter property requires some care, though, and here we provide two definitions that we study.

The notions are defined as distributions of the output of an adversary conditioned on a particular game. In said game, the adversary, with access to a public key, first outputs two messages (m_0 and m_1), following which he gets back an encryption of one of the messages. In one distribution the adversary is given an encryption of m_0 and in the other an encryption of m_1. As such the adversay can be seen as living in two "worlds", defined by the message which is given as a challenge, with the security statement being that the adversary cannot distinguish between these worlds. In [BDJR97] this is referred to as "left-or-right" indistinguishability.

The first, weaker definition is *chosen plaintext attack* (CPA) error. Informally, this property states that if an adversary is given query access to the random oracle and the public key, the adversary's output should not significantly differ whether the challenge is an encryption of m_0 or m_1. We define this formally.

Definition 5.2. *An encryption scheme* $\mathsf{ENC} := \mathsf{ENC}[\lambda, \ell, \ell_c]$ *in the random oracle model has* **CPA error** ϵ_{CPA} *if for every query bound* $t \in \mathbb{N}$ *for the random oracle, adversary size bound* $s \in \mathbb{N}$*, and* t*-query algorithm* A *of size* s*, the distributions* $\mathcal{D}^A_{\mathrm{CPA}}(0)$ *and* $\mathcal{D}^A_{\mathrm{CPA}}(1)$ *are* $\epsilon_{\mathrm{CPA}}(\lambda, \ell, t, s)$*-close, where:*

$$\mathcal{D}^A_{\mathrm{CPA}}(b) := \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^f(\mathsf{pk}) \\ \hat{c} \leftarrow \mathsf{ENC.Enc}^f(\mathsf{pk}, m_b) \\ \hat{b} \leftarrow A^f(\mathsf{aux}, \hat{c}) \end{array} \right\} \ .$$

The second, stronger definition, is *chosen ciphertext attack* (CCA) error. Informally, this property states that if an adversary is given query access to the random oracle, the public key, *and access to a decryption oracle*, the adversary's output should not significantly differ whether the challenge is an encryption of m_0 or m_1. We define this formally.

Definition 5.3. *An encryption scheme* $\mathsf{ENC} := \mathsf{ENC}[\lambda, \ell, \ell_c]$ *in the random oracle model has* **CCA error** ϵ_{CCA} *if for every query bound* $t \in \mathbb{N}$ *for the random oracle, query bound* $t_{\mathrm{DEC}} \in \mathbb{N}$ *for the decryption oracle, adversary size bound* $s \in \mathbb{N}$*, and* (t, t_{DEC})*-query admissible algorithm* A *of size* s*, the distributions* $\mathcal{D}^A_{\mathrm{CCA}}(0)$ *and* $\mathcal{D}^A_{\mathrm{CCA}}(1)$ *are* $\epsilon_{\mathrm{CCA}}(\lambda, \ell, t, t_{\mathrm{DEC}}, s)$*-close, where:*

$$\mathcal{D}^A_{\mathrm{CCA}}(b) := \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^{f, \mathsf{ENC.Dec}^f(\mathsf{sk}, \cdot)}(\mathsf{pk}) \\ \hat{c} \leftarrow \mathsf{ENC.Enc}^f(\mathsf{pk}, m_b) \\ \hat{b} \leftarrow A^{f, \mathsf{ENC.Dec}^f(\mathsf{sk}, \cdot)}(\mathsf{aux}, \hat{c}) \end{array} \right\} \ .$$

Above, A *is admissible if, after receiving* $\hat{c}$ *as input,* A *does not query* $\hat{c}$ *to the decryption oracle.*

Many encryption schemes that are CPA-secure can be broken if an adversary gains access to the decryption oracle. Although restricting the adversary from querying the decryption oracle on the challenge ciphertext might seem stringent, nothing prevents the adversary from querying the oracle on a strategically modified version of the challenge ciphertext. This method is commonly used to show that CPA-secure schemes are insecure under chosen ciphertext attacks (CCA). It intuitively highlights the necessity for a stronger *hiding* requirement in any CCA-secure encryption scheme.

Randomness of the algorithms. While $\mathsf{ENC.Dec}$ is a deterministic algorithm, we note that both $\mathsf{ENC.Gen}$ and $\mathsf{ENC.Enc}$ must be probabilistic for the basic security properties of an encryption scheme to hold. Indeed, if it were not the case, an adversary could simply trivially get the secret key by running $\mathsf{ENC.Gen}$ or use $\mathsf{ENC.Enc}$ to encrypt both of his messages and compare those with the challenge ciphertext, thereby always winning the experiment.

5.2 Construction

Given message length $\ell \in \mathbb{N}$, we define the relation

$$\mathcal{R}_\ell := \left\{ \Big((\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1), (\rho_0, m_0, \rho_1, m_1)\Big) : \begin{array}{l} m_0, m_1 \in \{0,1\}^\ell \\ \wedge\ m_0 = m_1 \\ \wedge\ c_0 = \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_0, m_0; \rho_0) \\ \wedge\ c_1 = \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_1, m_1; \rho_1) \end{array} \right\} . \tag{2}$$

Consider the following two ingredients: (i) a CPA-secure encryption scheme $\mathsf{ENC}_{\mathrm{CPA}} = (\mathsf{ENC.Gen}_{\mathrm{CPA}}, \mathsf{ENC.Enc}_{\mathrm{CPA}}, \mathsf{ENC.Dec}_{\mathrm{CPA}})$ with message length ℓ; and (ii) a non-interactive argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}_\ell$.

We will use the above ingredients to construct a CCA-secure encryption scheme ENC. Roughly speaking, the scheme adds a layer of complexity on top of the CPA scheme. A secret key is a tuple of two independently sampled CPA public/secret key pairs, and the corresponding public key is the tuple of the associated public keys (excluding the secret keys). Encrypting a message consists of obtaining two separate encryptions of the message under the two public keys, as well as sampling an argument string for the relation $\mathcal{R}_\ell$, attesting to the fact that these encryptions are of the same message. The encryption is the pair of CPA ciphertexts, along with the argument string. Decryption of a message consists of verifying that the argument string is accepting, followed by the decryption of one of the two ciphertexts using the CPA decryption algorithm.

Let $\lambda \in \mathbb{N}$ be a security parameter and $f \in \mathcal{U}(\lambda)$ be a random oracle. Each of the algorithms of ENC is constructed as follows.

- $\mathsf{ENC.Gen}(1^\lambda)$:
  1. Sample two independent key pairs $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{ENC.Gen}_{\mathrm{CPA}}(1^\lambda)$.
  2. Set the public key $\mathsf{pk} := (\mathsf{pk}_0, \mathsf{pk}_1)$.
  3. Set the secret key $\mathsf{sk} := (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1)$.
  4. Output the key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{ENC.Enc}^f\big(\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1), m \in \{0,1\}^\ell\big)$:
  1. For every $b \in \{0,1\}$, $c_b \leftarrow \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_b, m; \rho_b)$ for fresh encryption randomness ρ_b.
  2. Set the instance $\mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1)$.
  3. Set the witness $\mathbb{w} := (\rho_0, m, \rho_1, m)$.

4. Sample the argument string $\pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w})$.
5. Output the ciphertext $c := (c_0, c_1, \pi)$.

- $\mathsf{ENC.Dec}^f\big(\mathsf{sk} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1), c = (c_0, c_1, \pi)\big)$:
  1. Set the instance $\mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1)$.
  2. Check that $\mathcal{V}^f(\mathbb{x}, \pi) = 1$.
  3. Decrypt the first ciphertext: $m := \mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_0, c_0)$.
  4. Output the message m.

5.3 Security analysis

The security of the encryption scheme ENC constructed in Section 5.2 depends on several properties.

- The CPA security of $\mathsf{ENC}_{\mathrm{CPA}}$. Intuitively, if the hiding property of the underlying CPA encryption scheme were weak, then any adversary A with a non-negligible advantage in the CPA security game could be used to break the CCA security of the constructed scheme. Specifically, a CCA adversary trying to distinguish between two ciphertexts, given a CCA ciphertext constructed in the manner described above, can simply present one of the two CPA ciphertexts included in the challenge to the aforementioned "good" CPA adversary A and use his output to make his guess, thereby matching the CPA adversary's advantage.

- The zero-knowledge property of NARG. Encryptions in ENC (which are of the form (c_0, c_1, π)) contain within them argument strings that attest to statements of the form "there exists a message m and two random strings ρ_0 and ρ_1 such that for an instance $(\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1)$ it holds that $c_0 = \mathsf{ENC.Enc}_{\mathrm{CPA}}(\mathsf{pk}_0, m; \rho_0)$ and $c_1 = \mathsf{ENC.Enc}_{\mathrm{CPA}}(\mathsf{pk}_1, m; \rho_1)$". Hence, if NARG were not zero-knowledge then the argument string may in principle reveal m, or the encryption randomness, both of which permit for the computation of the message behind the challenge ciphertext.

- The simulation soundness property of NARG. Intuitively, the argument strings need to attest to the fact that both CPA ciphertexts are encryptions of the same message. As such, if coming up with an accepting argument string for a false statement were easy, an adversary could, in principle, manipulate his challenge ciphertext into one that can be queried to the decryption oracle simply by attaching an accepting argument string along with two ciphertexts that may not necessarily be encryptions of the same message. Therefore, if the first ciphertext comes from the challenge and the other is arbitrary, as long as the argument string is accepted, the adversary can get a decryption of his challenge ciphertext all while staying admissible.

  Furthermore, we emphasize that there is no requirement for the NARG to be simulation *knowledge* sound. In this adversarial setting, the existence of a witness is sufficient because the encryption algorithm is publicly available through the public key (i.e., it is not reserved for a party holding a secret). The witness is implicitly generated as a by-product of any honest encryption. Simulation soundness is, however, needed over plain soundness because an adversary capable of observing encryptions for different messages should not be able to gain any information about the underlying plaintext.

The following lemma formally captures the above intuition, as the CCA error is upper bounded by a sum of errors corresponding to the above properties.

Theorem 5.4. *Consider the following two ingredients:*

- $\mathsf{ENC}_{\mathrm{CPA}}$ *is an encryption scheme with CPA error* ϵ_{CPA}*; and*
- $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *is a non-interactive argument for the relation* $\mathcal{R}_\ell$ *(see Equation 2) with computational* true *simulation soundness error* $\epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}$ *and computational zero-knowledge error* z_{ARG}*.*

Furthermore, let:

- $t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}$ *be the number of random oracle queries made by* $\mathsf{ENC.Enc}^f_{\mathrm{CPA}}$*,*
- $t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}$ *be the number of random oracle queries made by* $\mathsf{ENC.Dec}^f_{\mathrm{CPA}}$*,*
- $t_{\mathrm{RO},\mathcal{V}}$ *be the number of random oracle queries made by* $\mathcal{V}$*,*
- $t_{\mathrm{RO},\mathcal{S}}$ *be the number of random oracle queries made by* $\mathcal{S}$*,*
- $\ell_{\mathsf{key},\mathrm{CPA}}$ *be the length of the CPA public key in the* $\mathsf{ENC}_{\mathrm{CPA}}$ *scheme; and*
- $\ell_{c,\mathrm{CPA}}$ *be the length of the CPA ciphertext in the* $\mathsf{ENC}_{\mathrm{CPA}}$ *scheme.*

Then for any random oracle query bound $t \in \mathbb{N}$*, decryption oracle query bound* $t_{\mathrm{DEC}} \in \mathbb{N}$ *and* (t, t_{DEC})*-query admissible adversary* A*,* $\mathsf{ENC} := \mathsf{ENC}[\lambda, \ell, \ell_c]$ *constructed in Section 5.2 is an encryption scheme that (is perfectly complete and) has CCA error* ϵ_{CCA} *such that*

$$\begin{aligned}
&\epsilon_{\mathrm{CCA}}(\lambda, \ell, t, t_{\mathrm{DEC}}, s) \leqslant \\
&\quad 2\big(z_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})) \\
&\quad + \epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})) \\
&\quad + \epsilon_{\mathrm{CPA}}(\lambda, \ell, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))\big) \ .
\end{aligned}$$

Proof. Let A be a (t, t_{DEC})-query admissible algorithm of size s, with respect to which the distributions $\mathcal{D}^A_{\mathrm{CCA}}(0)$ and $\mathcal{D}^A_{\mathrm{CCA}}(1)$ from Definition 5.3 are δ close. We argue that δ is upper bounded by the error expression given in the theorem statement.

We will proceed in a game hopping manner [Sho04], defining the necessary intermediate distributions, which differ only in the way that the encryption and decryption procedures are realized. We restate the distributions from Definition 5.3 in a manner that is specific to the construction in Section 5.2. Namely,

$$\mathcal{D}^A_{\mathrm{CCA}}(b) := \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^{f, \textsc{Decrypt}^f(\mathsf{sk}, \cdot, b)}(\mathsf{pk}) \\ \hat{c} \leftarrow \mathsf{ENC.Enc}^f(\mathsf{pk}, m_b) \\ \hat{b} \leftarrow A^{f, \textsc{Decrypt}^f(\mathsf{sk}, \cdot, b)}(\mathsf{aux}, \hat{c}) \end{array} \right\} = \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^{f, \textsc{Decrypt}^f(\mathsf{sk}, \cdot, b)}(\mathsf{pk}) \\ c_0 \leftarrow \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_0, m; \rho_0) \\ c_1 \leftarrow \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_1, m; \rho_1) \\ \mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1) \\ \mathbb{w} := (\rho_0, m, \rho_1, m) \\ \pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w}) \\ \hat{c} := (c_0, c_1, \pi) \\ \hat{b} \leftarrow A^{f, \textsc{Decrypt}^f(\mathsf{sk}, \cdot, b)}(\mathsf{aux}, \hat{c}) \end{array} \right\} ,$$

where the decryption oracle of the adversary, $\textsc{Decrypt}^f(\mathsf{sk}, \cdot, b_c)$, differs from $\mathsf{ENC.Dec}^f$ only in the fact that its third parameter, b_c, specifies which of the two CPA secret keys and ciphertexts should be used for the decryption. Since in both distributions both CPA encryptions are of the same message, which of the two keys is used for decryption bears no relevance. We provide the oracle pseudocode below.

$\underline{\textsc{Decrypt}^f(\mathsf{sk} := (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1), c := (c_0, c_1, \pi), b_c):}$
$\mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1)$
if $\mathcal{V}^f(\mathbb{x}, \pi) = 1$
  $m := \mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_{b_c}, c_{b_c})$
  return m
return $\perp$

We now define a series of hybrid distributions. Their main characteristic is that the argument strings are simulated by the zero-knowledge simulator $\mathcal{S}$, instead of the argument prover. As such, we replace the honest encryption algorithm $\mathsf{ENC.Enc}^f$ with $\textsc{Encrypt}^{f[\mu_{\mathrm{all}}]}_{\mathcal{S}}$ and parameterize the distributions with $b, b', b_c \in \{0,1\}$, where (1) the bits b and b' specify to the encryption algorithm that m_b and $m_{b'}$ are to be encrypted under pk_0 and pk_1, respectively; and (2) b_c specifies the ciphertext c_{b_c} and secret key sk_{b_c} to be used for decryption. We will refer to these as simulated world distributions.

$$\mathcal{D}^A_{\mathcal{S}}(b, b', b_c) := \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mu_{\mathrm{all}} := [\,] \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^{f[\mu_{\mathrm{all}}], \textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk}, \cdot, b_c)}(\mathsf{pk}) \\ (\hat{c}, \mu) \leftarrow \textsc{Encrypt}^{f[\mu_{\mathrm{all}}]}_{\mathcal{S}}(\mathsf{pk}, m_b, m_{b'}) \\ \mu_{\mathrm{all}} := \mu_{\mathrm{all}} \| \mu \\ \hat{b} \leftarrow A^{f[\mu_{\mathrm{all}}], \textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk}, \cdot, b_c)}(\mathsf{aux}, \hat{c}) \end{array} \right\}$$

$$= \left\{ \hat{b} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mu_{\mathrm{all}} := [\,] \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda) \\ (m_0, m_1, \mathsf{aux}) \leftarrow A^{f[\mu_{\mathrm{all}}], \textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk}, \cdot, b_c)}(\mathsf{pk}) \\ c_0 \leftarrow \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_0, m_b; \rho_0) \\ c_1 \leftarrow \mathsf{ENC.Enc}^f_{\mathrm{CPA}}(\mathsf{pk}_1, m_{b'}; \rho_1) \\ \mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1) \\ (\pi, \mu) \leftarrow \mathcal{S}^{f[\mu_{\mathrm{all}}]}(\mathbb{x}) \\ \hat{c} := (c_0, c_1, \pi) \\ \mu_{\mathrm{all}} := \mu_{\mathrm{all}} \| \mu \\ \hat{b} \leftarrow A^{f[\mu_{\mathrm{all}}], \textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk}, \cdot, b_c)}(\mathsf{aux}, \hat{c}) \end{array} \right\}.$$

We now use the above distributions to prove our first claim.

Claim 5.5 (Zero-knowledge bound)**.** *For any (t, t_{DEC})-query admissible adversary A, and $b \in \{0,1\}$, the distributions $\mathcal{D}^A_{\mathrm{CCA}}(b)$ and $\mathcal{D}^A_{\mathcal{S}}(b,b,b)$ are*

$$z_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO,Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO,Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))$$

-close.

We refer the reader to Appendix B.1 for the proof of Claim 5.5.

To proceed, we define a variant of the simulated world distribution, $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(b, b', b_c)$, which differs from $\mathcal{D}^A_{\mathcal{S}}(b, b', b_c)$ in that the adversary makes no *improper* queries to the decryption oracle. We define an *improper* ciphertext as $c := (c_0, c_1, \pi)$, where $\mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_0, c_0) \neq \mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_1, c_1)$, but where the verifier accepts π. In the claim below, we show that if the NARG is simulation sound, the two distributions must be close.

Claim 5.6 (Simulation soundness bound)**.** *For any (t, t_{DEC})-query admissible adversary A, and $b \in \{0,1\}$, the distributions $\mathcal{D}^A_{\mathcal{S}}(b,b,b)$ and $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(b,b,b)$ are*

$$\epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))$$

-close.

We refer the reader to Appendix B.2 for the proof of Claim 5.6.

Next, the claim below captures the fact that even if the challenge ciphertext is computed incorrectly, the adversary's behavior cannot change, or else it would contradict the CPA security of the underlying scheme.

Claim 5.7 (CPA security bound)**.** *For any (t, t_{DEC})-query admissible adversary, the distributions $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(0,0,0)$ and $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(0,1,0)$ are*

$$\epsilon_{\mathrm{CPA}}(\lambda, \ell, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))$$

-close. Analogously, the same holds for the distributions $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(0,1,1)$ and $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(1,1,1)$.

We refer the reader to Appendix B.3 for the proof of Claim 5.7.

Claim 5.8 (Decryption oracle change)**.** *For any (t, t_{DEC})-query admissible adversary, it holds that $\mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(0,1,0) \equiv \mathcal{D}^A_{\mathcal{S},\mathit{Proper}}(0,1,1)$.*

Proof. The equivalence of the two distributions trivially holds, since the adversary makes only proper queries to the decryption oracle, and therefore $\mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_0, c_0) = \mathsf{ENC.Dec}^f_{\mathrm{CPA}}(\mathsf{sk}_1, c_1)$. As a result, whether the decryption is done using sk_0 or sk_1 does not change the view of the adversary in the experiment. □

We obtain the following chain of distributions:

$$\begin{aligned}\mathcal{D}^A_{\mathrm{CCA}}(0) \longleftrightarrow \mathcal{D}^A_{\mathcal{S}}(0,0,0) \longleftrightarrow \mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0,0,0) \longleftrightarrow \mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0,1,0) \longleftrightarrow \\ \longleftrightarrow \mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0,1,1) \longleftrightarrow \mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(1,1,1) \longleftrightarrow \mathcal{D}^A_{\mathcal{S}}(1,1,1) \longleftrightarrow \mathcal{D}^A_{\mathrm{CCA}}(1)\ .\end{aligned}$$

Combining the bounds from Claims 5.5–5.8, we obtain that the distributions $\mathcal{D}^A_{\mathrm{CCA}}(0)$ and $\mathcal{D}^A_{\mathrm{CCA}}(1)$ are

$$\begin{aligned}2\big(&z_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})) \\ &+ \epsilon^{\mathrm{SIM}}_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})) \\ &+ \epsilon_{\mathrm{CPA}}(\lambda, \ell, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))\big)\end{aligned}$$

-close. This concludes the proof of the theorem.

□

5.4 Lower bounds

Similarly to Section 4.5, we discuss lower bounds related to the construction. We demonstrate a strict lower bound on the CPA security of the underlying scheme and discuss intuitions for lower bounds for the security properties of the underlying NARG.

Lower bound on CPA security. Let λ be the security parameter, and consider an adversary A_{CPA} of size $s \in \mathbb{N}$, with random oracle query bound t for whom the distributions $\mathcal{D}_{\mathrm{CPA}}^{A_{\mathrm{CPA}}}(0)$ and $\mathcal{D}_{\mathrm{CPA}}^{A_{\mathrm{CPA}}}(1)$ are at least $\epsilon_{\mathrm{CPA}}(\lambda, \ell, t, s)$-far. We can use this adversary to construct an adversary A that breaks the CCA security of ENC.

The attacker A receives access to the random oracle, and a decryption oracle. He works as follows:

$\underline{A^{f,\mathsf{ENC.Dec}(\mathsf{sk},\cdot)}\big(\mathsf{pk} := (\mathsf{pk}_0, \mathsf{pk}_1)\big)}$

1. Emulate the execution of $A_{\mathrm{CPA}}^{f}(\mathsf{pk}_0)$ appropriately relaying its queries to f.
2. When A halts with output (m_0, m_1, aux), output (m_0, m_1).
3. Upon receiving the CCA challenge ciphertext $\hat{c}_{\mathrm{CCA}} := (c_0, c_1, \pi)$:

   Emulate the execution of $A_{\mathrm{CPA}}^{f}(\mathsf{aux}, c_0)$ appropriately relaying its queries to f.
4. When A_{CPA} halts with output $\hat{b}$, return $\hat{b}$.

The attacker has size $s + O(1)$ where the additive constant comes from the instructions needed to simulate A_{CPA}. The adversary makes t queries to the random oracle, and 0 queries to the decryption oracle. As such for A, the distributions distinguishes between the distributions $\mathcal{D}_{\mathrm{CCA}}^{A}(0)$ and $\mathcal{D}_{\mathrm{CCA}}^{A}(1)$ are also $\epsilon_{\mathrm{CPA}}(\lambda, \ell, t, s)$-far. It follows that the CPA security of $\mathsf{ENC}_{\mathrm{CPA}}$ is a lower bound on the CCA security of ENC.

Lower bound on computational zero-knowledge. The underlying NARG in the construction satisfies the *multi-instance* zero-knowledge property. We notice that one-time computational zero-knowledge is sufficient to achieve chosen-ciphertext attack (CCA) security in our reduction, as at any step in the security game, the proving oracle is queried once, as demonstrated in the proof of Theorem 5.4.

The strictly weaker notion of witness indistinguishability, as defined in Definition 4.8, would be insufficient because, in general, we cannot assume that witnesses are unique.

We leave open the question of whether a further relaxation of the zero-knowledge property would be sufficient to prove the proposed scheme's adaptive chosen-ciphertext security (CCA-2).

Lower bound on computational *true*-simulation soundness. Firstly, similarly to the zero-knowledge property discussed above, it is sufficient for the construction to use a computational *true*-simulation sound NARG that holds against an attacker with a query budget of 1 to the simulation oracle.

Secondly, if the NARG was not simulation-sound, an adversary could maul its challenge ciphertext into one that he could query for the decryption oracle. However, similar issues arise as in Section 4.5, where, in particular, the simulation soundness property could be too general for the CCA-2 attack game (the CCA attacker is bound to a specific public key). We leave for future work the investigation of a concrete attack that showcases a tight lower bound on the requirement of this property.

6 Future work

We identify several areas of consideration for future work.

Different flavors of security. The signature and encryption schemes are proven secure based on well-accepted security definitions from the cryptography literature. Crucially, we consider bounded-size adversaries. For both primitives, it is advantageous to ensure security against query-bounded adversaries alone rather than against both query- and size-bounded adversaries. This new constraint would necessitate a different construction for the encryption scheme, as the CPA encryption scheme used as a building block may not generally withstand such adversaries.

Lower bounds. We have established some formal lower bounds and discussed the intuitive necessity of specific building blocks for the constructions we study. While relaxing specific properties would inevitably lead to weaker bounds, as demonstrated in Theorems 4.7 and 5.4, we acknowledge that even these relaxed properties could still be too strong for some practical applications, as noted in Sections 4.5 and 5.4. We, therefore, identify a direction for future research: obtaining tight lower bounds for both zero-knowledge and simulation security of the NARGs used in our constructions from Sections 4 and 5. Such an investigation could reveal more efficient constructions or uncover fundamental trade-offs between security guarantees and the strength of underlying assumptions.

Practical instantiations. The primary objective of this work was to formalize notions related to simulation security in the random oracle model and provide a concrete security treatment of the topic. As presented, we have explored various definitions, offering intuitive explanations for the relative strength of different notions. Sections 4.5 and 5.4, discuss the implications of relaxing the security of the building blocks. A natural extension of this theoretical foundation would be the practical instantiation of our schemes. This future work could involve implementing the schemes using concrete cryptographic building blocks and examining the nuances of parameter selection in practice.

Appendices

A Proof of Theorem 4.7

Let A be a (t, t_{SIG})-query admissible algorithm of size s that forges (wins the experiment in Definition 4.2) with probability δ. We argue that δ is upper bounded by the error expression given in the lemma statement. We proceed in three steps, each contributing an error term.

(1) Adversary against computational zero-knowledge. We define an attacker A_1 against the zero-knowledge property of NARG (Definition 2.8). The attacker receives query access to a random oracle and a proving oracle, which works as shown below.

$\underline{A_1^{f,\textsc{Prove}(\cdot,\cdot)}}$:

1. Sample an instance-witness pair : $(\mathbb{x}, \mathbb{w}) \leftarrow \mathsf{SIG.Gen}$.
2. Run $A(\mathbb{x})$ and answer each of his queries as follows:
   (a) Random oracle query x: return $y := f(x)$.
   (b) Signing oracle query m: return $\sigma := \textsc{Prove}\big((\mathbb{x}, m), \mathbb{w}\big)$.
3. When A terminates with an output (m, σ), return $((\mathbb{x}, m), \sigma)$.

The attacker A_1 makes at most t queries to the random oracle, makes at most t_{SIG} queries to the proving oracle, and queries instances of size at most $\lambda + \ell$. The attacker has size $s + \mathsf{poly}(\lambda, \ell)$, where the overhead comes from running $\mathsf{SIG.Gen}$. The attacker is admissible for the zero-knowledge property because his queries to the proving oracle are valid instance-witness pairs. By the zero-knowledge property of NARG, the statistical distance between the output of A_1 in the real-world experiment and the simulated-world experiment is at most $z_{\text{ARG}}(\lambda, t, t_{\text{SIG}}, \lambda + \ell, s + \mathsf{poly}(\lambda, \ell))$.

The attacker A_1 invokes A in such a way that A "sees" the unforgeability experiment when in the real-world experiment; in particular, A forges if and only if A_1 outputs $((\mathbb{x}, m), \sigma)$ such that $\mathcal{V}^f((\mathbb{x}, m), \sigma) = 1$ where $(\mathbb{x}, m)$ was not an instance queried to the proving oracle.

In the simulated-world experiment, each proving query $\big((\mathbb{x}, m), \mathbb{w}\big)$ is answered by the argument simulator $\mathcal{S}$ on input $(\mathbb{x}, m)$, which up to an error of $z_{\text{ARG}}(\lambda, t, t_{\text{SIG}}, \lambda+\ell, s+\mathsf{poly}(\lambda, \ell))$ is the same as in the real world experiment. In particular, the simulator is queried only on valid instance-witness pairs. Therefore, the probability that A_1 outputs $((\mathbb{x}, m), \sigma)$ such that $\mathcal{V}^{f[\mu_{\text{all}}]}((\mathbb{x}, m), \sigma) = 1$, where $(\mathbb{x}, m)$ was not an instance queried to the simulation oracle, is at least $\delta_1 := \delta - z_{\text{ARG}}(\lambda, t, t_{\text{SIG}}, \lambda + \ell, s + \mathsf{poly}(\lambda, \ell))$.

(2) Adversary against computational true-simulation knowledge soundness. We define an attacker A_2 against the simulation knowledge soundness property of NARG (Definition 3.3). The attacker receives query access to a random oracle and a simulation oracle, which he can query on true instances. The attacker works as follows:

$\underline{A_2^{f,\textsc{Sim}_{\mathsf{true}}(\cdot,\cdot)}}$:

1. Sample an instance-witness pair : $(\mathbb{x}, \mathbb{w}) \leftarrow \mathsf{SIG.Gen}$.
2. Run $A(\mathbb{x})$ and answer each of his queries as follows:
   (a) Random oracle query x: return $y := f(x)$.
   (b) Signing oracle query m: return $\sigma := \textsc{Sim}_{\mathsf{true}}\big((\mathbb{x}, m), \mathbb{w}\big)$.
3. When A terminates with an output (m, σ), return $\big((\mathbb{x}, m), \sigma\big)$.

The attacker A_2 makes at most t queries to the random oracle, makes at most t_{SIG} queries to the proving oracle, and queries instances of size at most $\lambda+\ell$. The attacker has size $s+\mathsf{poly}(\lambda,\ell)$, where the overhead comes from running $\mathsf{SIG.Gen}$. Moreover, A_2 invokes A in such a way that A "sees" the simulation security experiment, which is the same as the simulated-world experiment orchestrated by A_1 above (being admissible for the zero-knowledge property implies being admissible for the simulation security experiment). In particular, this means that up to an error of $z_{\text{ARG}}(\lambda, t, t_{\text{SIG}}, \lambda+\ell, s+\mathsf{poly}(\lambda,\ell))$ adversary "sees" the unforgeability experiment.

Hence, by the simulation knowledge soundness property, the probability that $\mathcal{E}((\mathbb{x}, m), \sigma, \mathsf{tr}_{\text{RO}})$ outputs $\mathbb{w}$ such that $((\mathbb{x}, m), \mathbb{w}) \in \mathcal{R}_\ell$ is at least $\delta_2 := \delta_1 - \kappa^{\text{SIM}}_{\text{ARG}}(\lambda, t, t_{\text{SIG}}, \lambda+\ell, s+\mathsf{poly}(\lambda,\ell))$.

(3) Adversary against hard relation. We define an attacker A_3 against the hardness of $\mathcal{R}$ (Definition 4.5). The attacker receives as input a challenge instance $\mathbb{x}$ sampled as $(\mathbb{x}, \mathbb{w}) \leftarrow G(1^\lambda)$, and works as follows.

$\underline{A_3(\mathbb{x})}$:

1. Lazily sample a random oracle $f \leftarrow \mathcal{U}(\lambda)$.
2. Initialize an empty query-answer list μ_{all}.
3. Emulate an execution of $A(\mathbb{x})$ and do the following to answer his queries.
   (a) Random oracle query x: answer with $y := f[\mu_{\text{all}}](x)$.
   (b) Signing oracle query $m \in \{0,1\}^\ell$: sample $(\pi, \mu) \leftarrow \mathcal{S}^{f[\mu_{\text{all}}]}((\mathbb{x}, m))$, append μ to μ_{all} provided that μ and μ_{all} do not share queries, and answer with π. (Abort if μ and μ_{all} share queries.)
4. When A terminates with an output (m, σ), run $\mathbb{w} \leftarrow \mathcal{E}((\mathbb{x}, m), \sigma, \mathsf{tr}_{\text{RO}})$, and return $\mathbb{w}$.

The attacker A_3 has size at most $s+\mathsf{poly}(\lambda, \ell, t, t_{\text{SIG}})$, where the additive cost in addition to the size s of A comes from the lazy simulation of the random oracle and its programming, the simulation of the signing oracle, as well as from running the argument extractor $\mathcal{E}$. Moreover, A_3 invokes A so that A "sees" the simulation security experiment, as is the previous point. Hence, the probability that $\mathcal{E}$ outputs a valid witness is at least δ_2. However, by the hardness of the relation $\mathcal{R}_\ell$, we know that $\delta_2 \leqslant \epsilon_{\mathcal{R}}(\lambda+\ell, s+\mathsf{poly}(\lambda, \ell, t, t_{\text{SIG}}))$, which concludes the theorem. □

B Proofs of the claims supporting Theorem 5.4

Below we provide proofs for Claims 5.5–5.7.

B.1 Proof of Claim 5.5

We define an attacker A_1 against the zero-knowledge property of NARG, which attempts to distinguish between the distributions $\mathcal{D}^A_{\text{CCA}}(b)$ and $\mathcal{D}^A_{\mathcal{S}}(b,b,b)$. The attacker receives query access to a random oracle and a proving oracle, and works as below.

$\underline{A_1^{f,\textsc{Prove}(\cdot)}}$:

1. Sample a public and secret key pair: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{ENC.Gen}(1^\lambda)$.
2. Run $A^{f,\textsc{Decrypt}^f(\mathsf{sk},\cdot,b)}(\mathsf{pk})$ and answer each of his queries as follows:
   (a) Random oracle query x: return $y := f(x)$.
   (b) Decryption oracle query $c := (c_0, c_1, \pi)$: return $m := \textsc{Decrypt}^f(\mathsf{sk}, c, b)$.

3. When A halts with output (m_0, m_1, aux), compute the challenge ciphertext: $\hat{c} \leftarrow \textsc{Encrypt}^f_{\textsc{Prove}}(\mathsf{pk}, m_b)$.
4. Resume the execution of $A^{f,\textsc{Decrypt}^f(\mathsf{sk},\cdot,b)}(\mathsf{aux}, \hat{c})$, and return his output b'.

For each query A makes to the decryption oracle, A_1 queries the random oracle $t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}$ times. When A_1 computes the challenge ciphertext, he queries the random oracle $2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}$ times. Therefore, A_1 will make at most $t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}$ queries to the random oracle.

A_1 makes exactly one query to the proving oracle, when computing the challenge ciphertext. The queries to the proving oracle are made on instances of size at most $2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}$. A_1 has size at most $s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})$, where the overhead comes from running the $\mathsf{ENC.Gen}$, $\textsc{Encrypt}$ and $\textsc{Decrypt}$ procedures. We note that the attacker is admissible for the zero-knowledge property because he only queries the proving oracle on valid instance-witness pairs.

When A_1 is in the real world experiment, $\textsc{Encrypt}^f_{\textsc{Prove}}$ corresponds to $\textsc{Encrypt}^f_{\mathcal{P}}$. In particular, the random oracle in not programmed, and the honest prover answers with a real argument string. When A_1 is in the simulated world experiment, $\textsc{Encrypt}^f_{\textsc{Prove}}$ corresponds to $\textsc{Encrypt}^{f[\mu_{\mathrm{all}}]}_{\mathcal{S}}$, which involves programming of the random oracle.[3] We emphasize, that the simulation oracle in this case corresponds to $\textsc{Sim}^{f[\mu_{\mathrm{all}}]}_{\mathsf{true}}$, since all proving queries are on true instances.

In the real-world experiment, A_1 simulates A in such a way that the distribution of his output is exactly $\mathcal{D}^A_{\mathrm{CCA}}(b)$. In the simulated-world experiment, A_1 simulates A_1 is such a way, that the distribution of his output is exactly $\mathcal{D}^A_{\mathcal{S}}(b, b, b)$.

It follows from the zero-knowledge property, that $\mathcal{D}^A_{\mathrm{CCA}}(b)$ and $\mathcal{D}^A_{\mathcal{S}}(b, b, b)$ are

$$z_{\mathrm{ARG}}(\lambda, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}}, 1, 2\ell_{\mathsf{key},\mathrm{CPA}} + 2\ell_{c,\mathrm{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})) \qquad (3)$$

-close. □

B.2 Proof of Claim 5.6

We define an attacker A_2 against the simulation soundness property of NARG (Definition 3.3 and **??**). The attacker has access to a random oracle, and a simulation oracle $\textsc{Sim}_{\mathsf{true}}$ from which he can obtain simulated argument strings for true instances. A_2 simulates A in the same way as A_1 does in Claim 5.5 when in the simulated world experiment (i.e.: $\textsc{Encrypt}^f_{\textsc{Prove}}$ corresponds to $\textsc{Encrypt}^{f[\mu_{\mathrm{all}}]}_{\mathcal{S}}$, and the simulation oracle is queried only on valid instance-witness pairs). The additional difference is the following: if at any point A makes an accepting but improper query to the decryption oracle, A_2 halts his execution (and that of A) and returns this query. That is, whenever A queries the decryption oracle, A_2 runs the following alternative to the $\textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk}, \cdot, b_c)$ as a subroutine.

[3] We highlight that in the simulated-world experiment, the random oracle is programmed in a way that is unknown to A. In particular, we note that, it is possible for A to notice an inconsistency, in the scenario where the simulator programs the oracle at a position x, and A queries the random oracle at x before *and* after receiving the challenge. This case is however already part of the zero-knowledge error bound.

$\underline{\textsc{Decrypt}^{f[\mu_{\text{all}}]}_{\mathsf{Proper}}(\mathsf{sk} := (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{sk}_1), c := (c_0, c_1, \pi), b_c):}$

$\mathbb{x} := (\mathsf{pk}_0, c_0, \mathsf{pk}_1, c_1)$
if $\mathcal{V}^{f[\mu_{\text{all}}]}(\mathbb{x}, \pi) = 1$:
  if $\mathsf{ENC.Dec}^{f[\mu_{\text{all}}]}_{\text{CPA}}(\mathsf{sk}_0, c_0) \neq \mathsf{ENC.Dec}^{f[\mu_{\text{all}}]}_{\text{CPA}}(\mathsf{sk}_1, c_1)$:
    "Halt, and output c."
  $m := \mathsf{ENC.Dec}^{f[\mu_{\text{all}}]}_{\text{CPA}}(\mathsf{sk}_{b_c}, c_{b_c})$
  return m
return $\perp$

For each query A makes to the decryption oracle, A_2 queries the random oracle $t_{\text{RO},\mathcal{V}} + 2t^{\text{CPA}}_{\text{RO,Dec}}$ times. When A_2 computes the challenge ciphertext, he queries the random oracle $2t^{\text{CPA}}_{\text{RO,Enc}}$ times. Therefore, A_2 will make at most $t + t_{\text{DEC}} \cdot (t_{\text{RO},\mathcal{V}} + 2t^{\text{CPA}}_{\text{RO,Dec}}) + 2t^{\text{CPA}}_{\text{RO,Enc}}$ queries to the random oracle.

A_2 makes exactly one query to the simulation oracle, when computing the challenge ciphertext, on an instance of size at most $2\ell_{\mathsf{key},\text{CPA}} + 2\ell_{c,\text{CPA}}$. A_2 has size at most $s + \mathsf{poly}(\lambda, \ell, t, t_{\text{DEC}})$, where the overhead comes from running the $\mathsf{ENC.Gen}$, Encrypt and Decrypt procedures.

It follows, that if A makes no improper, but accepting queries, the distributions $\mathcal{D}^A_{\mathcal{S}}(b, b, b)$ and $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(b, b, b)$ are identical. On the other hand, if A ever makes an improper but accepting query, then A_2 would successfully intercept (and subsequently output) a ciphertext containing a valid argument string for a false statement. By the simulation soundness property of NARG, this can happen with probability at most

$$\epsilon^{\text{SIM}}_{\text{ARG}}(\lambda, t + t_{\text{DEC}} \cdot (t_{\text{RO},\mathcal{V}} + 2t^{\text{CPA}}_{\text{RO,Dec}}) + 2t^{\text{CPA}}_{\text{RO,Enc}}, 1, 2\ell_{\mathsf{key},\text{CPA}} + 2\ell_{c,\text{CPA}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\text{DEC}})),$$

which subsequently bounds the statistical distance between the distributions. □

B.3 Proof of Claim 5.7

We prove here the statistical closeness of the distributions $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0, 0, 0)$ and $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0, 1, 0)$. The proof for the distributions $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(0, 1, 1)$ and $\mathcal{D}^A_{\mathcal{S},\mathsf{Proper}}(1, 1, 1)$ is analogous.

We define an attacker A_3 against the CPA security of $\mathsf{ENC}_{\text{CPA}}$ (Definition 5.2). The attacker receives query access to a random oracle and receives a CPA public key pk_1 as input, sampled using $\mathsf{ENC.Gen}_{\text{CPA}}$. The attacker outputs two messages during his execution, following which he receives a challenge ciphertext, which is the CPA encryption of one of the messages. He then carries on with his execution and terminates by outputting a bit b'. Formally the adversary works as follows.

$\underline{A_3(\mathsf{pk}_1)}$:

1. Sample a CPA public and secret key pair: $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{ENC.Gen}_{\text{CPA}}(1^\lambda)$.
2. Set $\mathsf{pk} := (\mathsf{pk}_0, \mathsf{pk}_1)$, $\mathsf{sk} := (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \perp)$.
3. Run $A^{f, \textsc{Decrypt}^f(\mathsf{sk}, \cdot, 0)}(\mathsf{pk})$ and answer each of his queries as follows:
   (a) Random oracle query x: return $y := f(x)$.
   (b) Decryption oracle query $c := (c_0, c_1, \pi)$: return $m := \textsc{Decrypt}^f(\mathsf{sk}, c, 0)$.
4. When A halts with output (m_0, m_1, aux), output (m_0, m_1).
5. Upon receiving the CPA challenge ciphertext $\hat{c}_{\text{CPA}}$:
   (a) Compute $c_0 \leftarrow \mathsf{ENC.Enc}^f_{\text{CPA}}(m_0; \rho_0)$; and
   (b) Sample a simulated argument string $(\pi, \mu) \leftarrow \mathcal{S}^{f[\mu_{\text{all}}]}\big((\mathsf{pk}_0, c_0, \mathsf{pk}_1, \hat{c}_{\text{CPA}})\big)$.

6. Set $\hat{c} := (c_0, \hat{c}_{\mathrm{CPA}}, \pi)$, and resume the execution of $A^{f[\mu_{\mathrm{all}}],\textsc{Decrypt}^{f[\mu_{\mathrm{all}}]}(\mathsf{sk},\cdot,0)}(\mathsf{aux}, \hat{c})$.
7. When A halts with output b', return b'.

For each query A makes to the decryption oracle, A_3 queries the random oracle $t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}$ times. When A_3 computes the challenge ciphertext, he queries the random oracle $2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}$ times. Therefore, A_3 will make at most $t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}$ queries to the random oracle.

The attacker A_3 has size $s+\mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}})$, where the polynomial overhead comes from running the argument simulator as well as simulating the decryption and encryption procedures. We note that A_3 has no trouble simulating the decryption oracle for A, as he uses sk_0 to decrypt. We also note that in the second phase of A's execution, A simulates the random oracle in a way that is consistent with the programming μ.

We can observe, that when the challenge ciphertext is an encryption of m_0, the distribution of the output of A_3 is exactly $\mathcal{D}^{A}_{\mathcal{S},\mathsf{Proper}}(0,0,0)$. On the other hand, when the challenge ciphertext is an encryption of m_1, the distribution of the output of A_3 is exactly $\mathcal{D}^{A}_{\mathcal{S},\mathsf{Proper}}(0,1,0)$. It follows that the two distributions are

$$\epsilon_{\mathrm{CPA}}(\lambda, \ell, t + t_{\mathrm{DEC}} \cdot (t_{\mathrm{RO},\mathcal{V}} + t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Dec}}) + 2t^{\mathrm{CPA}}_{\mathrm{RO},\mathsf{Enc}} + t_{\mathrm{RO},\mathcal{S}}, s + \mathsf{poly}(\lambda, \ell, t, t_{\mathrm{DEC}}))$$

-close, because in the contrary case, it would imply that A_3 can distinguish between the two cases, breaking the CPA security of $\mathsf{ENC}_{\mathrm{CPA}}$. □

References

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. "Simultaneous Hardcore Bits and Cryptography against Memory Attacks". In: *Theory of Cryptography*. Ed. by Omer Reingold. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 474–495. ISBN: 978-3-642-00457-5 (cit. on p. 19).

[Bab85] L Babai. "Trading group theory for randomness". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, pp. 421–429. ISBN: 0897911512. DOI: 10.1145/22145.22192. URL: https://doi.org/10.1145/22145.22192 (cit. on p. 6).

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Theory of Cryptography*. Ed. by Martin Hirt and Adam Smith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 31–60. ISBN: 978-3-662-53644-5 (cit. on p. 6).

[BDFLSZ11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random oracles in a quantum world". In: *Proceedings of ASIACRYPT 2011 (17th International Conference on the Theory and Application of Cryptology and Information Security)*. ASIACRYPT 2011. 2011, pp. 41–69. URL: https://doi.org/10.1007/978-3-642-25385-0_3 (cit. on p. 6).

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. "A concrete security treatment of symmetric encryption". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE. 1997, pp. 394–403 (cit. on pp. 7, 26).

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking computations in polylogarithmic time". In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. STOC '91. New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, pp. 21–32. ISBN: 0897913973. DOI: 10.1145/103418.103428. URL: https://doi.org/10.1145/103418.103428 (cit. on p. 6).

[BG93] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *Advances in Cryptology — CRYPTO' 92*. Ed. by Ernest F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 390–420. ISBN: 978-3-540-48071-6 (cit. on pp. 6, 11).

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions". In: *Advances in Cryptology — CRYPT0' 95*. Ed. by Don Coppersmith. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 15–28. ISBN: 978-3-540-44750-4 (cit. on p. 7).

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. "The Security of Cipher Block Chaining". In: *Advances in Cryptology — CRYPTO '94*. Ed. by Yvo G. Desmedt. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 341–358. ISBN: 978-3-540-48658-9 (cit. on p. 7).

[BR93] Mihir Bellare and Phillip Rogaway. "Random oracles are practical: a paradigm for designing efficient protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. CCS '93. Fairfax, Virginia, USA: Association for Computing Machinery, 1993, pp. 62–73. ISBN: 0897916298. DOI: 10.1145/168588.168596. URL: https://doi.org/10.1145/168588.168596 (cit. on pp. 6, 10).

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. "The random oracle methodology, revisited". In: *J. ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411. DOI: 10.1145/1008731.1008734. URL: https://doi.org/10.1145/1008731.1008734 (cit. on p. 6).

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *Theory of Cryptography*. Ed. by Dennis Hofheinz and Alon Rosen. Cham: Springer International Publishing, 2019, pp. 1–29. ISBN: 978-3-030-36033-7 (cit. on p. 6).

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. URL: https://snargsbook.org (cit. on pp. 6, 8, 9, 14, 16).

[DDN00] Dvora Dolev, C Dwork, and Moni Naor. "Nonmalleable cryptography". eng. In: *SIAM Journal on Computing* 30.2 (2000), p. 47. ISSN: 0097-5397. DOI: 10.1137/S0097539795291562 (cit. on pp. 7, 8, 15).

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-malleable cryptography". In: (1991), pp. 542–552 (cit. on pp. 7, 8, 15).

[DDOPS01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. "Robust Non-interactive Zero Knowledge". In: *Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 566–598. ISBN: 978-3-540-44647-7 (cit. on p. 7).

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model". In: *Advances in Cryptology – CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Cham: Springer International Publishing, 2019, pp. 356–383. ISBN: 978-3-030-26951-7 (cit. on p. 6).

[DH76] Whitfield Diffie and Martin E Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976) (cit. on p. 20).

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. "Efficient Public-Key Cryptography in the Presence of Key Leakage". In: *Advances in Cryptology - ASIACRYPT 2010*. Ed. by Masayuki Abe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 613–631. ISBN: 978-3-642-17373-8 (cit. on pp. 7, 8, 16, 19, 20).

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. "On the Non-malleability of the Fiat-Shamir Transform". In: *Progress in Cryptology - INDOCRYPT 2012*. Ed. by Steven Galbraith and Mridul Nandi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 60–79. ISBN: 978-3-642-34931-7 (cit. on pp. 7, 8, 19).

[FS86] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 186–194. ISBN: 978-3-540-47721-1 (cit. on pp. 6, 8).

[FS89] Uriel Feige and Adi Shamir. "Zero knowledge proofs of knowledge in two rounds". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 526–544 (cit. on pp. 6, 11).

[FS90] Uriel Feige and Adi Shamir. "Witness indistinguishable and witness hiding protocols". In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. STOC '90. Baltimore, Maryland, USA: Association for Computing Machinery, 1990, pp. 416–426. ISBN: 0897913612. DOI: 10.1145/100216.100272. URL: https://doi.org/10.1145/100216.100272 (cit. on p. 23).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption". In: vol. 28. 2. Elsevier, 1984, pp. 270–299 (cit. on pp. 7, 25).

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof-systems". In: STOC '85 (1985), pp. 291–304. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178 (cit. on pp. 6, 11, 13).

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". In: *SIAM Journal on computing* 17.2 (1988), pp. 281–308 (cit. on p. 19).

[GMY06] Juan A. Garay, Philip MacKenzie, and Ke Yang. "Strengthening Zero-Knowledge Protocols Using Signatures". In: *Journal of Cryptology* 19.2 (Apr. 2006), pp. 169–209. ISSN: 1432-1378. DOI: 10.1007/s00145-005-0307-3. URL: https://doi.org/10.1007/s00145-005-0307-3 (cit. on p. 7).

[JP11] Abhishek Jain and Omkant Pandey. *Non-Malleable Zero Knowledge: Black-Box Constructions and Definitional Relationships*. Cryptology ePrint Archive, Paper 2011/513. https://eprint.iacr.org/2011/513. 2011. URL: https://eprint.iacr.org/2011/513 (cit. on pp. 7, 8, 15).

[KV09] Jonathan Katz and Vinod Vaikuntanathan. "Signature Schemes with Bounded Leakage Resilience". In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 703–720. ISBN: 978-3-642-10366-7 (cit. on pp. 7, 19).

[LZ19] Qipeng Liu and Mark Zhandry. "Revisiting Post-quantum Fiat-Shamir". In: *Advances in Cryptology – CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Cham: Springer International Publishing, 2019, pp. 326–355. ISBN: 978-3-030-26951-7 (cit. on p. 6).

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000), pp. 1253–1298. DOI: 10.1137/S0097539795284959. eprint: https://doi.org/10.1137/S0097539795284959. URL: https://doi.org/10.1137/S0097539795284959 (cit. on p. 6).

[Nat] National Institute of Standards and Technology. *Cryptographic Algorithm Validation Program*. https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program. Computer Security Resource Center (cit. on p. 6).

[NY90] Moni Naor and Moti Yung. "Public-key cryptosystems provably secure against chosen ciphertext attacks". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 427–437 (cit. on pp. 7, 25).

[PR05] Rafael Pass and Alon Rosen. "New and improved constructions of non-malleable cryptographic protocols". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 533–542. ISBN: 1581139608. DOI: 10.1145/1060590.1060670. URL: https://doi.org/10.1145/1060590.1060670 (cit. on pp. 7, 8).

[Sah99] A. Sahai. "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security". en. In: *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. New York City, NY, USA: IEEE Comput. Soc, 1999, pp. 543–553. ISBN: 978-0-7695-0409-4. DOI: 10.1109/SFFCS.1999.814628. URL: http://ieeexplore.ieee.org/document/814628/ (visited on 04/20/2024) (cit. on pp. 7, 8, 15).

[Sch90] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 239–252. ISBN: 978-0-387-34805-6 (cit. on pp. 6, 12).

[Sho04] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Paper 2004/332. https://eprint.iacr.org/2004/332. 2004. URL: https://eprint.iacr.org/2004/332 (cit. on p. 29).

[YZ22] Takashi Yamakawa and Mark Zhandry. "Verifiable quantum advantage without structure". In: *Proceedings of FOCS 2022 (63rd Symposium on Foundations of Computer Science)*. FOCS 2022. 2022, pp. 69–74. URL: https://doi.org/10.1109/FOCS54457.2022.00014 (cit. on p. 6).