Rogue-Instance Security for Batch Knowledge Proofs

Gil Segev* Amit Sharabi† Eylon Yogev‡

September 20, 2023

Abstract

We propose a new notion of knowledge soundness, denoted *rogue-instance security*, for interactive and non-interactive *batch* knowledge proofs. Our notion, inspired by the standard notion of rogue-key security for multi-signature schemes, considers a setting in which a malicious prover is provided with an honestly-generated instance $\mathbb{x}_1$, and may then be able to maliciously generate related "rogue" instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ for convincing a verifier in a batch knowledge proof of corresponding witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$ for all k instances – without actually having knowledge of the witness $\mathbb{w}_1$ corresponding to the honestly-generated instance. This setting provides a powerful security guarantee for batch versions of a wide variety of practically-relevant protocols, such as Schnorr's protocol and similar ones.

We present a highly-efficient generic construction of a batch proof-of-knowledge applicable to any *algebraic* Sigma protocols. The algebraic property refers to a homomorphic structure of the underlying group and includes Schnorr's protocol and others. We provide an almost tight security analysis for our generic batch protocol, which significantly improves upon the previously known security bounds even for the specific case of batch Schnorr protocol. We extend our results beyond algebraic Sigma protocols. We analyze the rogue-instance security of a general batch protocol with plus-one special soundness (a generalization of standard special soundness) and achieve improved security bounds in the generic case.

Our results use a particular type of *high-moment* assumptions introduced by Rotem and Segev (CRYPTO 2021). These assumptions consider the hardness of a relation against algorithms with bounded *expected* running time. Although Rotem and Segev introduced these assumptions, they did not provide evidence to support their hardness. To substantiate and validate the high-moment assumptions, we present a new framework for assessing the concrete hardness of cryptographic problems against oracle algorithms with bounded expected runtime. Our framework covers generic models, including the generic group model, random oracle model, and more. Utilizing our framework, we achieve the first hardness result for these high-moment assumptions. In particular, we establish the second-moment hardness of the discrete-logarithm problem against expected-time algorithms in the generic group model.

*School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: segev@cs.huji.ac.il. Supported by the Israel Science Foundation (Grant No. 1336/22) and by the European Union (ERC, FTRC, 101043243). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

†Department of Computer Science, Bar-Ilan University, Israel. Email: amit.sharabi1@live.biu.ac.il. Sponsored by the Israel Science Foundation (Grant No. 2439/20).

‡Department of Computer Science, Bar-Ilan University, Israel. Email: eylon.yogev@biu.ac.il. Eylon Yogev is supported by an Alon Young Faculty Fellowship, by the Israel Science Foundation (Grant No. 2302/22), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

Contents

1 Introduction

A zero-knowledge proof-of-knowledge protocol is a powerful cryptographic tool with diverse applications. It enables a prover to convincingly demonstrate to a verifier, who holds an instance $\mathbb{x}$, that it possesses knowledge of a valid witness $\mathbb{w}$ for $\mathbb{x}$. The fundamental power of such protocols lies in the ability to *extract* a witness from a given prover, a property that varies in its precise formulation across different protocols. Proofs of knowledge play a pivotal role in cryptographic protocols, both from a theoretical standpoint and in practical implementations.

One notable example is Schnorr's protocol [Sch89, Sch91], which serves as a zero-knowledge proof-of-knowledge for the knowledge of the discrete-logarithm of a group element. In its interactive form, this protocol offers an efficient identification scheme, while in its non-interactive form, it translates into a signature scheme via the Fiat-Shamir transformation. The widespread influence of the Schnorr identification and signature schemes stems from their conceptual simplicity and practical efficiency. Another compelling example is a proof-of-knowledge for a Pedersen commitment or hash function, which is the product of two Schnorr instances. In this scenario, the prover demonstrates the ability to "open" the commitment without actually revealing its contents, thus maintaining the privacy of the committer [Oka92]. The wide-ranging applicability of these protocols within the field of cryptography has garnered substantial attention and interest in a tight analysis of their security bounds.

Extraction from special soundness. Both of the examples presented above exemplify Sigma protocols, which are three-move protocols that exhibit the unique soundness notion called "special soundness". This property plays a vital role in the construction of an extractor. Specifically, the property states that it is possible to extract a witness when provided with two accepting transcripts that share the same first message but differ in the second message. Consequently, to establish the protocol's security based on the hardness of the underlying relation, the extractor must successfully extract two such valid transcripts from a potentially malicious prover.

To achieve this goal, existing approaches employ a strategy of executing the protocol multiple times. The analysis of these approaches draws upon the classic "forking lemma" introduced by Pointcheval and Stern [PS00] (see also [AAB$^+$02, BN06, BCC$^+$16, KMP16]). These different approaches showcase a trade-off between the success probability and the running time of the extractor. To provide a concrete example, let us examine the Schnorr identification scheme and signature scheme, which derive their security from the hardness of the discrete-logarithm problem. For the Schnorr identification scheme, suppose we have a malicious prover who runs in time t and succeeds in impersonating with probability ϵ. We can transform this malicious prover into a discrete-logarithm algorithm that runs in time $2t$ and succeeds with probability ϵ^2. Similarly, for the Schnorr signature scheme, suppose the attacker additionally performs at most q queries to the random oracle. We can transform this attacker into a discrete-logarithm algorithm that runs in time $2t$ and succeeds with probability ϵ^2/q. For any group of order p, where generic hardness of discrete-log is believed to hold [Sho97], this leads to the bound $\epsilon \leq (t^2/p)^{1/2}$ for the Schnorr identification scheme, and a bound of $\epsilon \leq (q \cdot t^2/p)^{1/2}$ for the Schnorr signature scheme. Other trade-offs that were established lead to the same bound [BD20, JT20]. In idealized models, such as the generic group model [Sho97, Mau05] and the algebraic group model [FKL18, AHK20, BFL20, FPS20, MTT19, RS20], it is possible to achieve an optimal bound of $\epsilon \leq t^2/p$ (see [Sho97, FPS20]).

High-moment forking lemma. The extractor runs the given adversary for the second time, only if the first time succeeded. Thus, it is convenient to analyze the *expected* running-time of the extractor, rather than its strict running-time [KL08]. In this case, the result is an algorithm for solving discrete-logarithm with a bound on its expected running time. Recently, Segev and Rotem [RS21] have leveraged this type of analysis to derive tighter bounds for Schnorr's protocols (and similar Sigma protocols). Towards this end, they established a hardness of discrete-logarithm for excepted time algorithms.

In simple terms, their *second-moment assumption* states that the success probability ϵ of any algorithm A solving discrete-logarithm for a group of order p satisfies $\epsilon \leq \mathbb{E}\left[T_A^2\right]/p$, where T_A denotes the random

variable corresponding to A's running time.[1] Under this assumption, Segev and Rotem were able to derive the bound of $\epsilon \leq (t^2/p)^{2/3}$, which is the best-known bound for Schnorr in the standard model. Achieving the optimal bound in the standard model remains an open problem that continues to drive ongoing research and exploration.

Batch protocols. The Schnorr protocol and the Pedersen protocol both admit efficient *batch* versions [GLS+04]. A batch protocol is given k instances, $\mathbb{x}_1, \ldots, \mathbb{x}_k$, and allows to prove the knowledge of *all* corresponding k witnesses with a communication complexity that is approximately the same as that of a single proof of knowledge. The efficiency gain provided by batch protocols is a highly desirable property in many domains. In the context of blockchain, batching is a widely adopted practice aimed at reducing costs and optimizing resource utilization, the instances are usually public-keys and the witnesses are private-keys. By grouping multiple transactions or operations into a single batch, the associated overhead, such as communication and computation costs (which affect the transaction fees), can be significantly reduced.

However, the security analysis of batch protocols raises several concerns. The security bounds vary depending on how the instances are chosen in the security game (a modeling issue that does not appear with a single instance). For example, in a permissionless blockchain network, the attacker can choose the instances (its public-keys) adaptively as a function of existing instances sampled by honest parties. In such a case, the security reduction cannot assume hardness of the instances chosen by the adversary. These types of security games are known in the context of multi-signatures and are called rogue-key attacks (see [BN06, BDN18, MPS+19, BD21, NRS21] and the many references therein).

The special soundness property extends to the multiple instance case. In this setting, the extractor must extract $k+1$ valid transcripts from which it can compute all k corresponding witnesses (actually, it needs all $k+1$ transcripts even if it aims to compute a single witness). This is a generalization of the standard special soundness property, which we call *plus-one* special soundness. However, deriving tight security bounds for the batch setting is even more challenging than the single case. A straightforward extension of the single extractor to the batch version would run the malicious prover $k+1$ times and would yield an extractor that runs in approximately $(k+1) \cdot t$ time, but with a success probability of ϵ^{k+1}, i.e., an exponential decay in the number of instances. This is indeed the case in the batch Schnorr protocol given in [GLS+04]. Furthermore, the tighter bound of Segev and Rotem [RS21] does not seem to extend to the multiple instance case (regardless of the precise security game definition). This raises the question of how to derive tight security bounds for batch protocols.

1.1 Our contributions

We give several contributions towards a better understanding of batch proof-of-knowledge protocols.

Rogue-instance soundness. Our first contribution is a strong security notion for batch protocols, denoted *rogue-instance security*, for interactive and non-interactive *batch* knowledge proofs. Our notion is inspired by the standard notion of rogue-key security for multi-signature schemes. We consider a setting in which a malicious prover is provided with an honestly-generated instance $\mathbb{x}_1$ (according to some distribution), and is then allowed to maliciously generate related "rogue" instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ for convincing a verifier in a batch knowledge proof of corresponding witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$ for all k instances. This is done without the malicious prover having knowledge of the witness $\mathbb{w}_1$ corresponding to the honestly-generated instance. This setting provides a powerful security guarantee for batch versions of numerous practical and relevant protocols, such as Schnorr's protocol and similar ones. See Section 4 for the precise definition.

Batching algebraic Sigma protocols We construct batch protocols for a large family of Sigma protocols and provide a relatively tight analysis. Our construction works for *algebraic* Sigma protocols, which captures the proof-of-knowledge protocol for discrete-logarithm (Schnorr) [Sch89, Sch91], Pedersen commitment [Oka92], Guillou-Quisquater identification scheme [GQ88] and more. The algebraic property refers to a

[1] They originally stated their assumption for a general d-moment but, in this paper, we focus on the second-moment.

homomorphic structure of the underlying group. Algebraic Sigma protocols consist of an algebraic one-way function f such that the prover aims to prove knowledge of a preimage under f. The notion of algebraic one-way function introduced by Catalano et al. [CFG+15] which relates to the notion of group-homomorphic one-way generators introduced by Cramer and Damgård [CD98]. We analyze the security of our construction in the rogue-instance game and achieve the bound $\epsilon \leq (t^2/p)^{2/3}$ (for groups of order p) which matches the state-of-the-art bound of Segev and Rotem [RS21] for a single instance. In particular, our bound does not depend on the number of rogue instances. In more general form, our theorem is as follows.

Theorem 1 (Informal)**.** *Let* Π *be an algebraic Sigma protocol for a relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$*. If* $\mathcal{R}$ *is second-moment hard with respect to a distribution* $\mathcal{D}$*, then* $\mathcal{R}$ *has a batch protocol with rogue soundness error* $\epsilon(t) \leq (t^2/|\mathcal{W}|)^{2/3}$*.*

In particular, our theorem gives us tighter security bounds for the batch version of Schnorr and Pederson protocols. Specifically, the batch version of Schnorr's protocols immediately implies the same bounds for the corresponding batch identification scheme.

Corollary 1.1. *Assuming that the discrete-logarithm problem is second-moment hard, any adversary that runs in time* t *wins in the rogue soundness game for the batch Schnorr and Okamoto identification schemes with probability at most* $(t^2/p)^{2/3}$*, where* p *is the order of the underlying group.*

We extend our results for general batch Sigma protocols. We analyze the rogue-instance security of a general batch protocol with plus-one special soundness and achieve the bound of $\epsilon \leq (k^2 \cdot t^2/p)^{1/2}$, which is inferior to our bound for the specific case of algebraic protocols, but superior to previously known bounds.

Theorem 2 (Informal)**.** *Let* Π *be* k*-batch Sigma protocol for a relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ *with plus-one special soundness. If* $\mathcal{R}$ *is second-moment hard with respect to a distribution* $\mathcal{D}$*, then* Π *has rogue soundness error* $\epsilon(t) \leq (k^2 \cdot t^2/|\mathcal{W}|)^{1/2}$*.*

In Table 1 we exemplify the concrete improvements we get in Theorem 1 and Theorem 2 for various parameter settings.

Attacker's running time t	**Security parameter** λ	**Batch parameter** k	**Bound of [GLS+04]** $(t^2/p)^{1/(k+1)}$	**Generic bound Theorem 2** $(k^2 \cdot t^2/p)^{1/2}$	**Algebraic bound Theorem 1** $(t^2/p)^{2/3}$
2^{64}	256	2	$2^{-42.67}$	2^{-63}	$2^{-85.33}$
2^{64}	256	4	$2^{-25.6}$	2^{-62}	$2^{-85.33}$
2^{80}	256	6	$2^{-13.71}$	$2^{-45.42}$	2^{-64}
2^{80}	512	8	$2^{-39.11}$	2^{-173}	$2^{-234.66}$
2^{100}	512	16	$2^{-18.35}$	2^{-152}	2^{-208}
2^{100}	512	24	$2^{-12.48}$	$2^{-151.42}$	2^{-208}
2^{128}	512	24	$2^{-10.24}$	$2^{-123.42}$	$2^{-170.66}$
2^{128}	512	32	$2^{-7.76}$	2^{-123}	$2^{-170.66}$

Table 1: A comparison of the security guarantees for the batch Schnorr scheme provided by [GLS+04] compared to our bounds given in Theorem 2 and in Theorem 1.

Non-interactive proof-of-knowledge. We construct non-interactive batch arguments from algebraic Sigma protocols by applying the Fiat-Shamir paradigm to the batch Sigma protocols. Given Theorem 1, the generic analysis of the Fiat-Shamir yields a bound on the rogue-instance game of $\epsilon \leq q \cdot (t^2/p)^{2/3}$ when considering malicious prover who runs in time t and performs at most q queries to the random oracle. However, direct analysis of the rogue-instance game yields a bound of $\epsilon \leq (kq \cdot t^2/p)^{2/3}$ which is again matches the bound of Rotem and Segev [RS21], for a single instance. Informally, we show the following:

Theorem 3 (Informal)**.** *Let* Π *be an algebraic Sigma protocol for a relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$*. If* $\mathcal{R}$ *is second-moment hard with respect to a distribution* $\mathcal{D}$*, then* $\mathcal{R}$ *has a non-interactive batch argument with rogue soundness error* $\epsilon(t) \leq (kq \cdot t^2/|\mathcal{W}|)^{2/3}$*.*

Establishing hardness for high-moment assumptions. Theorem 1 and Theorem 3 rely on the second-moment-hardness of a relation, an assumption introduced in [RS21]. While the use of these assumptions is beneficial, there is no evidence to support their hardness. To remedy the situation, we present a new framework that allows to establish bounds for oracle-algorithms with expected running time. Utilizing our framework, we achieve the first hardness result for these high-moment assumptions, relative to a oracle. The general statement of our framework is somewhat technical and is given in Theorem 9.5. Thus, we present two main implications of our framework, which are easier to state.

First, we establish the second-moment hardness of the discrete-logarithm problem against expected-time algorithms in the generic group model. Shoup [Sho97] analyzed the generic hardness of the discrete-logarithm problem with respect to strict time algorithms. He showed that any generic t-time algorithm that solves the discrete-logarithm problem has success probability at most $\epsilon \leq t^2/p$. Applying our framework yields a bound of $\epsilon \leq \mathbb{E}\left[T_A^2\right]/p$ when considering *unbounded* algorithms where T_A denotes the random variable indicating the algorithm's running time.

Theorem 4 (second-moment hardness in generic group model; Informal)**.** *For any query algorithm* A*, let* $T_A = T_A(\lambda)$ *be a random variable indicating the number of queries performed by* A *until he stops. For every algorithm* A *that solves the discrete-logarithm problem in a generic group of prime order* p *and succeeds with probability* ϵ_A *it holds that*

$$\epsilon_A \leq \frac{\mathbb{E}\left[T_A^2\right]}{p} \ .$$

Our framework is inspired by [JT20] which showed a generic framework to prove bounds with respect to expected-time algorithms when considering only the first-moment of the expected running time. Their result proves the first-moment assumption (Definition 3.1), but cannot be used to derive second-moment hardness. Moreover, our framework achieves tighter bounds than theirs and is arguably easier to use (see Corollary 9.6).

Second, we derive expected-time bounds for SNARKs in the random oracle model (ROM). We focus on the construction of Micali [Mic00], which compiles a PCP to a SNARK in the ROM. It is known that if the underlying PCP has soundness error ϵ_{PCP}, then every malicious prover that makes at most t-queries to the random oracle can convince the verifier of a false statement with probability at most $\epsilon \leq t \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{t^2}{2^\lambda}$ (see analysis in [BCS16]). Using our framework, we derive the following bound.

Theorem 5 (second-moment hardness of SNARKs; Informal)**.** *Suppose the Micali construction is instantiated for a relation* $\mathcal{R}$ *with a PCP with error* ϵ_{PCP}*, and random oracle with output length* λ*. Then, for every* $\mathbb{x} \notin \mathcal{L}(\mathcal{R})$ *and every malicious argument prover* $\tilde{\mathcal{P}}$ *that performs* $T_{\tilde{\mathcal{P}}}$ *oracle queries (as a random variable) and outputs a proof* $\tilde{\pi}$ *it holds that*

$$\Pr\left[\mathcal{V}^f(\mathbb{x}, \tilde{\pi}) = 1\right] \leq \mathbb{E}\left[T_{\tilde{\mathcal{P}}}\right] \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{\mathbb{E}\left[T_{\tilde{\mathcal{P}}}^2\right]}{2^\lambda} \ .$$

In Section 2.6, we further discuss the type of cryptographic problems relative to an oracle captured by our framework. A formal treatment of the framework, including definitions, statements, and further examples, is given in Section 9.1.

2 Our techniques

We summarize the main ideas behind our results.

- In Section 2.1 we discuss the computational assumptions we consider in this work.
- In Section 2.2 we define batch Sigma protocols and extend the notion of rogue-key security for multi-signature, to rogue-instance security of batch proof-of-knowledge.
- In Section 2.3 we first show a general compiler from a large family of Σ-protocols to a batch Σ-protocol. Then, we show the high-level proof of the rogue-security of batch Σ-protocols constructed via the general compiler.
- In Section 2.4 we start by showing how to construct non-interactive batch arguments using the general compiler, then, we bound their rogue-security.
- In Section 2.5 we show how to apply our techniques on a general batch Σ-protocol and derive a concrete bound on their rogue-soundness error.
- In Section 2.6 we describe our framework for establishing high-moment hardness assumptions.

2.1 High-moment hardness

We begin by describing the computational assumptions that underlie our work. Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a relation, where $\mathcal{X}$ is the set of instances and $\mathcal{W}$ is the set of witnesses. We note that the relation (and in fact all algorithms that will be described later on) are with respect to a setup algorithm that produces public parameters. For the simplicity of this high-level overview, we omit the public parameters (where formal definitions take them into account).

We consider distribution $\mathcal{D}$ over instance-witness pairs such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. For example, the distribution can sample a discrete-logarithm challenge. Typically, the hardness of the distribution is stated with respect to strict-time algorithms, that is, algorithms that run in some fixed time t. Here, we consider hardness with respect to an algorithm where the running time, t, is a random variable. We denote by $T_{\mathcal{A},\mathcal{D}}$ the random variable indicating the running time of A on input $\mathbb{x}$ where $(\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}$. Informally, we say that $\mathcal{R}$ is *first-moment hard* with respect to the distribution $\mathcal{D}$ if for every algorithm A, it holds that

$$\textbf{first-moment hardness:} \quad \Pr\left[(\mathbb{x}, A(\mathbb{x})) \in \mathcal{R}\right] \leq \frac{\mathbb{E}\left[T_{A,\mathcal{D}}\right]}{|\mathcal{W}|^{0.5}} \ , \tag{1}$$

where the probability is taken over $(\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}$ and over A. The first-moment assumption is justified by the work of Jaeger and Tessaro [JT20]. They developed a framework for proving tight bounds on the advantage of an adversary with expected-time guarantees in generic models (a.k.a. "bad flag analysis"). In particular, they prove the first-moment hardness of the discrete-logarithm problem in the generic group model. That is, they show that every algorithm A with an expected running time $\mathbb{E}\left[T_A\right]$ computes the discrete-logarithm problem in the generic group model with probability at most $\mathbb{E}\left[T_A\right]/p^{1/2}$ (where p is the group size).

Recently, Rotem and Segev [RS21] have generalized this assumption for higher moments, where most important for our work is the second-moment assumption. We say that a relation is *second-moment hard* with respect to a distribution $\mathcal{D}$ if for every algorithm A it holds that

$$\textbf{second-moment hardness:} \quad \Pr\left[(\mathbb{x}, A(\mathbb{x})) \in \mathcal{R}\right] \leq \frac{\mathbb{E}\left[T^2_{A,\mathcal{D}}\right]}{|\mathcal{W}|} \ , \tag{2}$$

where the probability is taken over $(\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}$ and the algorithm A. The hardness of the second-moment assumption does not follow from the framework of [JT20], and has no justification even in generic models. In order to validate this assumption, we develop a framework (see Section 2.6), in the spirit of [JT20] which *does* allow us to establish bounds for second-moments. In particular, it allows us to prove the second-moment hardness of the discrete-logarithm problem in the generic group model. That is, we show that every algorithm A with an expected running time $\mathbb{E}\left[T_A\right]$ computes the discrete-logarithm problem in the generic group model with probability at most $\mathbb{E}\left[T^2_A\right]/p$.

2.2 Rogue-instance security for batch protocols

We move on to describe our notion of rogue-instance soundness for batch protocols. In a batch Σ-protocol, we are given k instance-witness pairs $(\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)$. The prover consists of two algorithms $\mathbf{P} = (\mathbf{P}_1, \mathbf{P}_2)$, where $\mathbf{P}_1$ sends a message α, the verifier $\mathbf{V}$ sends a random challenge $\beta \in \mathcal{C}$, $\mathbf{P}_2$ responds with a message γ, and the verifier $\mathbf{V}$ decides whether to accept.

The standard adaptive soundness requirement considers the case where a malicious prover wishes to convince the verifier on k instances of its choice. However, we consider batch Σ-protocols with rogue-instance security, where one instance $\mathbb{x}_1$ is sampled according to a given hard distribution, and the rest of the instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ are chosen adaptively as a function of $\mathbb{x}_1$.

Specifically, a batch Σ-protocol Π has ϵ rogue-soundness error if for every malicious prover $\tilde{\mathbf{P}} = (\tilde{\mathbf{P}}_1, \tilde{\mathbf{P}}_2)$ that runs in time t it holds that

$$\Pr\left[\mathsf{RogueExp}_{\Pi}(\tilde{\mathbf{P}}, \lambda) = 1\right] \leq \epsilon(t) \ ,$$

where the experiment $\mathsf{RogueExp}_{\Pi}(\tilde{\mathbf{P}}, \lambda)$ defined as follows:

1. $(\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda$
2. $((\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k), \alpha, \mathsf{st}) \leftarrow \tilde{\mathbf{P}}_1(\mathbb{x}_1)$
3. $\beta \leftarrow \mathcal{C}$
4. $\gamma \leftarrow \tilde{\mathbf{P}}_2(\mathsf{st}, \beta)$
5. Output $\mathbf{V}(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \beta, \gamma)$.

Recall that the definition above omits the setup phase, see Section 4 for the precise definition.

2.3 Batching algebraic Sigma protocols

We first describe our general compiler for batching algebraic Σ-protocols. This compiler takes an algebraic protocol (which we define next) and outputs a batch version of it (for the same relation). Then, we show the high-level proof of our (almost tight) rogue-security for the batch protocol.

Algebraic Sigma protocols. Algebraic Σ-protocols are defined with respect to an algebraic one-way function F. The protocol is a proof-of-knowledge of a preimage of $\mathsf{F}(r)$, for randomly sampled r. It is a generalization of the *preimage protocol* presented by Cramer and Damgård [CD98]. Algebraic one-way functions were introduced by Catalano et al. [CFG+15], a closely related notion to group-homomorphic one-way functions introduced by [CD98].

Informally, we say that a one-way function $\mathsf{F}\colon \mathcal{A}^m \to \mathcal{B}$ is algebraic if $\mathcal{A}$ and $\mathcal{B}$ are abelian cyclic groups and for every $x, x' \in \mathcal{A}^m$ it holds that $\mathsf{F}(x + x') = \mathsf{F}(x) \cdot \mathsf{F}(x')$. We say that a Σ-protocol $\Pi = (\mathbf{P}_1, \mathbf{P}_2, \mathbf{V})$ is algebraic if the protocol has the following general recipe:

1. The prover $\mathbf{P}_1$ produces a message $\alpha = \mathsf{F}(r)$ for $r \in \mathcal{A}$.
2. A challenge β is sampled from $\mathbb{Z}_p$ where p is the order of $\mathcal{A}$.
3. The prover $\mathbf{P}_2$ produces a message $\gamma = r + \beta \cdot \mathbb{w}$.
4. The verifier checks correctness by checking whether $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \mathbb{x}^\beta$.

General compiler to batch Sigma protocols. We construct a batch Σ protocol $\Pi^* = (\mathbf{P}_\mathbf{1}^*, \mathbf{P}_\mathbf{2}^*, \mathbf{V}^*)$ from algebraic Σ-protocol by invoking the Σ-protocol k times. Specifically, given k instances, $\mathbf{P}_\mathbf{1}^*$ invokes $\mathbf{P}_1(\mathbb{x}_i)$ and produces the message α which is the multiplication of all α_i's. Then, given k challenges, $\mathbf{P}_\mathbf{2}^*$ invokes $\mathbf{P}_2$ for each challenge and produces the compressed message γ by summing the messages γ_i. More formally, given an algebraic Σ-protocol $\Pi = (\mathbf{P}_1, \mathbf{P}_2, \mathbf{V})$, we construct a batch Σ-protocol $\Pi^* = (\mathbf{P}_\mathbf{1}^*, \mathbf{P}_\mathbf{2}^*, \mathbf{V}^*)$ as follows:

1. The prover $\mathbf{P}_\mathbf{1}^*$ invokes $\alpha_i \leftarrow \mathbf{P}_1(\mathbb{x}_i)$ and produces the message $\alpha = \Pi_{i=1}^k \alpha_i$.
2. k challenges β_i are sampled from $\mathbb{Z}_p$ where p is the order of $\mathcal{A}$.

3. The prover $\mathbf{P}_\mathbf{2}^*$ invokes $\gamma_i \leftarrow \mathbf{P}_2(\beta_i)$ for each challenge β_i and produces the compressed message $\gamma = \sum_{i=1}^{k} \gamma_i$.
4. The verifier checks correctness by checking whether $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \Pi_{i=1}^{k} \mathbb{x}_i^{\beta_i}$.

One can observe that the completeness of Π^* follows from the homomorphic property of F. The prover-to-verifier communication is two group elements. The verifier sends k elements, but since they are all uniformly random strings, they can be easily compressed to a single group element using any pseudo-random generator (e.g., using a random oracle).

Our objective is now to bound the rogue-soundness error of Π^*. To achieve this, we consider a malicious prover $\tilde{\mathbf{P}}$ that given as input an instance $\mathbb{x}_1$ which is sampled from a distribution $\mathcal{D}$, and chooses the rest of the instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ as a function of $\mathbb{x}_1$. Its goal is to convince the verifier on $\mathbb{x}_1, \ldots, \mathbb{x}_k$. We construct an algorithm that given as input an instance $\mathbb{x}$, invokes $\tilde{\mathbf{P}}$ on $\mathbb{x}$ in order to obtain a witness for $\mathbb{x}$. Combined with the second-moment assumption, it allows us to bound $\tilde{\mathbf{P}}$'s success probability (which is the rogue-soundness error).

In order to construct A, we make use of the special soundness property of Σ-protocols. Note that if a Σ-protocol has special soundness, then our construction yields a batch protocol which has *plus-one* special soundness (i.e., given $k+1$ accepting transcripts on k instances with a common first message and pairwise distinct challenges, one can extract all k witnesses). Obtaining $k+1$ valid transcripts from the adversary is very costly. However, in our case, we are only interested in extracting a single witness. Thus, we define a relaxed notion called *local special soundness* that allows to extract a single witness from two specifically designed transcripts.

Local special soundness. Informally, a batch Σ-protocol has *local special soundness* if there exists an extractor E such that given k instances $\mathbb{x}_1, \ldots, \mathbb{x}_k$ and a pair of accepting transcripts with a common first message and only one different challenge $\beta_i \neq \beta_i'$, outputs a valid witness for $\mathbb{x}_i$. We now show that every batch Σ-protocol constructed from algebraic Σ-protocol as above, has local special soundness.

Claim 1 (Informal)**.** *The batch Σ-protocol Π^* constructed above from algebraic Σ-protocol has local special soundness.*

Proof sketch. Consider the algorithm E which takes as input a pair of accepting transcripts $(\alpha, \beta_1, \ldots, \beta_k, \gamma)$, $(\alpha, \beta_1', \ldots, \beta_k', \gamma')$ such that there exists only one index j on which $\beta_j \neq \beta_j'$, defined as follows:

1. Let i^* be the index on which $\beta_{i^*} \neq \beta_{i^*}'$.
2. Output $(\gamma - \gamma')/(\beta_{i^*} - \beta_{i^*}')$ on the group $\mathbb{Z}_p$ where p is the order of $\mathcal{A}_{\mathsf{pp}}$.

The proof follows from the homomorphic property of F (see Section 5.1 for a complete proof). □

Due to the local special soundness property, it is sufficient to construct an algorithm A that invokes $\tilde{\mathbf{P}}$ on $\mathbb{x}$ and outputs two accepting transcripts $(\alpha, \beta_1, \ldots, \beta_k, \gamma), (\alpha, \beta_1', \ldots, \beta_k', \gamma')$ such that $\beta_1 \neq \beta_1'$.

We reduce the problem of finding two such transcripts to the "collision game" first introduced in [Cra96]. In more detail, we show that given an algorithm that succeeds in the collision game, we can construct an algorithm that outputs two such transcripts, which conclude extracting a witness.

The collision game. We consider the collision game first introduced in [Cra96] and used in [ACK21, HL10] which consists of a binary matrix $H \in \{0,1\}^{R \times N}$. The output of the game is 1 if and only if two 1-entries in the same row have been found.

Informally, the R rows correspond to the prover's randomness and the N columns correspond to the verifier's randomness. An entry of H equals 1 if and only if the corresponding transcript is accepting. Then, finding two 1-entries in the same row corresponds to finding two accepting transcripts with a common first message and distinct challenges. Therefore, an algorithm for the collision game can be transformed into an algorithm that finds two accepting transcripts, which by the local special soundness, allows extracting a witness (see Section 5.3 for a complete proof).

We now focus on constructing an algorithm for the collision game. In contrast to the collision game algorithm of [Cra96] which runs in strict polynomial time, our algorithm runs in expected polynomial time. A similar approach can be found in [ACK21, HL10], however, their algorithm minimizes only the first-moment of the expected running time. The collision game algorithm of [ACK21, HL10] samples an entry of H, if this entry equals 1, the algorithm continues to sample the entire row till it finds another 1-entry. One can observe that the second-moment of the expected running time of this algorithm is too high to get improved bounds.

Our goal is to construct an algorithm that maximizes the trade-off between the success probability and the second-moment of the expected running time, in order to use the second-moment assumption.

Lemma 1 (Informal)**.** *Let $H \in \{0,1\}^{R \times N}$ be a binary matrix and let ϵ be the fraction of 1-entries in H. Then, there exists an algorithm A with oracle access to H such that the following holds:*

1. *The expected number of queries performed by A to H is at most 2.*
2. *The second-moment of the expected number of queries performed by A to H is at most 4.*
3. *The probability that A succeeds in the collision game is at least $\epsilon^{1.5}$.*

Proof sketch. Let $B = \frac{1}{\sqrt{\epsilon}}$ and consider the following algorithm A:

A^H

1. Sample an entry (ρ, β) in H. If $H[\rho, \beta] = 0$, abort. Let $F = \emptyset$.
2. For every $i \in [B]$: sample without replacement entries in the same row ρ. If $H[\rho, \beta_i] = 1$, set $F \leftarrow F \cup \{\beta_i\}$.
3. If $F = \emptyset$, abort. Otherwise, choose uniformly at random an index $\beta' \in F$ and output ρ, β, β'.

Let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to H. For this section only, we omit the bound on the expected number of queries and refer to the second-moment only. A complete proof of the formal lemma can be found in Section 5.2.

By the description of A it performs 1 query to H with probability $(1 - \epsilon)$ and $(1 + B)$ queries with probability ϵ. Therefore,

$$\mathbb{E}\left[\mathcal{Q}_A^2\right] = (1-\epsilon) \cdot 1^2 + \epsilon \cdot (1+B)^2 \leq 1 + 2\sqrt{\epsilon} + 1 \leq 4 \ .$$

For now, we give a high-level overview of the proof of A's success probability. A complete proof can be found in Section 5.2. Assuming the first query to H was 1-entry, the algorithm continues to sample entries in the same row. Thus, if it hit a row with only one 1-entry, it succeeds in the game with probability zero. Therefore, we divide the rows by the number of 1-entries in it and look at the probability to sample such a row. Formally, for every $0 \leq d \leq N$, we let δ_d be the fraction of rows with exactly d 1-entries. Assuming the first query was 1-entry, A succeeds in the game if it finds at least one more 1-entry with B draws. Let X_d be a random variable indicating the number of 1-entries found in B draws in a row with exactly d 1-entries. Overall,

$$\Pr\left[\mathsf{CollGame}(A, H) = 1\right] \geq \sum_{d=2}^{N} \delta_d \cdot \frac{d}{N} \cdot \Pr\left[X_d \geq 1\right] \ .$$

In Section 5.2, we show that the above term is bounded by $\approx \epsilon^{1.5}$. □

2.4 Non-interactive batch arguments

In the previous subsection we showed a general compiler for batching algebraic Σ-protocols and bound their rogue-soundness error. Similarly, in this subsection we refer to the non-interactive analog. We first construct non-interactive batch arguments from algebraic Σ-protocols and then bound their rogue-instance security.

Non-interactive batch arguments from Sigma protocols. We show how to construct non-interactive batch arguments from algebraic Σ-protocols.

The construction is given by applying the Fiat-Shamir paradigm on the batch Σ-protocol constructed in Section 2.3 except for one minor change. Recall that in the construction of batch Σ-protocols, the prover is given as input k different challenges for each input. We wish to keep this property in the non-interactive analog. Specifically, we construct a non-interactive batch argument $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ from algebraic Σ-protocol by invoking the Σ-protocol k times and obtaining the challenges from a random oracle function $f \in \mathcal{U}(\lambda)$. In more detail, given k instances, the prover $\mathcal{P}$ invokes $\alpha_i \leftarrow \mathbf{P}_1(\mathbb{x}_i)$ and computes α as the multiplication of α_i's. Then, it obtains each challenge β_i by querying $f(\mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, i)$. Finally, it invokes $\mathbf{P}_2$ for each challenge and computes γ by summing the messages γ_i. The prover $\mathcal{P}$ outputs the proof string (α, γ). The verifier $\mathcal{V}$ computes β_i by querying the random oracle f and checking whether $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \Pi_{i=1}^{k} \mathbb{x}_i^{\beta_i}$. One can observe that the completeness of NARG follows from the homomorphic property of F and that the proof size is two group elements.

Our objective now is to bound the rogue-soundness error of NARG. Similarly to the interactive case, the NARG constructed above has local special soundness. Therefore, in order to extract a witness, it suffices to construct an algorithm that outputs a pair of transcripts with a common first message and only one different challenge $\beta_i \neq \beta_i'$.

Collision game for the non-interactive analog. Similar to the interactive case, our goal is to reduce the task of finding two such transcripts to the collision game. However, this transformation presents certain challenges. First, in the interactive case, we have two elements of randomness - the prover's randomness and the verifier's randomness which can be straightforwardly represented as a matrix. In contrast, in the non-interactive settings, the verifier's randomness is replaced by random oracle queries. A malicious prover performs at most q queries to the random oracle in order to obtain the challenges. Each answer from the random oracle may affect the prover's algorithm.

Secondly, in the interactive case, a prover $\mathbf{P}$ can be represented by two algorithms $\mathbf{P}_1, \mathbf{P}_2$. The algorithm $\mathbf{P}_1$ outputs the first message α and a state st, and $\mathbf{P}_2$ given as input the challenges β_i and the state st. Consequently, in order to obtain a pair of transcripts with a common first message, we can invoke $\mathbf{P}_1$ and $\mathbf{P}_2$, followed by invoking $\mathbf{P}_2$ again, on the same state and different challenges. In the non-interactive analog, a prover $\mathcal{P}$ outputs the instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ along with (α, γ). We assume without loss of generality that $\mathcal{P}$ always outputs α that it queried the random oracle f with $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha)$. Then, in order to obtain two transcripts with a common first message, we need to "guess" which random oracle query the prover is going to output. We invoke the prover once to obtain $(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \gamma)$ and let i^* be the random oracle on which the prover queried $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha)$. Then, we invoke the prover, replicating the same random oracle responses up to the i^*-th query. With probability $\approx 1/\mathsf{q}$ the prover outputs the same instances and first message α.

Therefore, we reduce the problem of finding two such transcripts into the "tree game". In this game, we consider a fixed randomness for the prover and consider a tree of depth q and degree 2^λ. The depth corresponds to the number of queries performed by the prover and the degree corresponds to the possible answers from the random oracle f. Consequently, the execution of the prover corresponds to a random walk on the tree and a leaf corresponds to the output of the prover. We let the value of a leaf be the random oracle query on which the prover queried f with this output. More precisely, each leaf corresponds to an output $(\mathbb{x}_2, \ldots, \mathbb{x}_k, \alpha, \gamma)$, we consider the value of a leaf to be the random oracle query in which the prover queried f with $(\mathbb{x}_2, \ldots, \mathbb{x}_k, \alpha)$. Then, finding two transcripts with a common first message and distinct challenges corresponds to finding two leaves with the same value i such that their lowest common ancestor is an internal node v of height i. A formal proof of the reduction can be found in Section 6.3.

The tree game. We introduce a tree game where an algorithm is given oracle access to a tree T where the value of each leaf is a number. Consider a complete tree T of depth l and degree r. Let $\mathsf{Leaves}(T)$ be the leaves of T and for every $u \in \mathsf{Leaves}(T)$ let $\mathsf{val}(u)$ be the value "stored" in u. Note that not all leaves hold a number value, we consider the value of such a leaf as $\perp$. During the execution of the game, the algorithm A

is given as input a number k and oracle access to the tree T and aims to find $k+1$ leaves $u_1, \ldots, u_{k+1}$ with the same value i that have the same lowest common ancestor v such that $\mathsf{height}(v) = i$.

Due to the local special soundness property, it is sufficient to construct an algorithm that outputs two accepting transcripts, then in this section, we consider the specific case where $k = 1$.

Lemma 2 (Informal)**.** *Let T be a complete tree of depth l and degree r and let ϵ be the fraction of non-bot leaves in T. Then, there exists an algorithm A with oracle access to T such that on input $k = 1$ the following holds:*

1. *The expected number of queries performed by A to H is at most 2.*
2. *The second-moment of the expected number of queries performed by A to H is at most 4.*
3. *The probability that A succeeds in the collision game is at least $\epsilon^{1.5}/l$.*

Proof sketch. Let $B = \frac{1}{\sqrt{\epsilon}}$ and consider the following algorithm A:

A^T

1. Sample a leaf $u \in \mathsf{Leaves}(T)$. If $\mathsf{val}(u) = \perp$, abort.
2. Let v be the parent of u of height $\mathsf{val}(u)$ and let w be the parent of u of height $(\mathsf{val}(u) - 1)$. Let $F = \emptyset$.
3. For every $i \in [B]$: sample without replacement leaves from $T_v \setminus T_w$. If $\mathsf{val}(u_i) = \mathsf{val}(u)$, set $F \leftarrow F \cup \{u_i\}$.
4. If $F = \emptyset$, abort. Otherwise, choose uniformly at random a leaf $u' \in F$ and output u, u'.

Let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to T. For this section only, we omit the bound on the expected number of queries and refer to the second-moment only. A complete proof of the formal lemma can be found in Section 6.2.

By the description of A it performs 1 query to T with probability $(1 - \epsilon)$ and $(1 + B)$ queries with probability ϵ. Therefore,

$$\mathbb{E}\left[\mathcal{Q}_A^2\right] = (1-\epsilon) \cdot 1^2 + \epsilon \cdot (1+B)^2 \leq 1 + 2\sqrt{\epsilon} + 1 \leq 4 \ .$$

For now, we give an informal high-level overview of the proof of A's success probability. A complete proof can be found in Section 6.2. Assume A samples a leaf u with the value h, then, A continues to sample leaves from the same sub-tree in order to find another leaf with the value h. Let v be the parent of u of height h. Note that for every h and v, the number of leaves with the value h in T_v may be different, which affects its success probability. Therefore, for every value h, we "divide" the internal nodes to "buckets" by the probability to sample a leaf with the value h in its sub-tree, and then we look at the probability to "reach" each bucket.

Formally, for every $0 \leq d \leq l \log r$ and $0 \leq h \leq l - 1$, we let

$$\delta_{d,h} = \Pr_{v:\mathsf{height}(v)=h}\left[\frac{|\{u \in \mathsf{Leaves}(T_v) : \mathsf{val}(u) = h\}|}{|\mathsf{Leaves}(T_v)|} \in \left[2^{-d}, 2^{-d+1}\right]\right] \ .$$

Note that a node v is in the d-th "bucket" if the probability to sample a leaf with the value h in the sub-tree T_v is in $\left[2^{-d}, 2^{-d+1}\right]$. Assuming the first query to the tree is a leaf u with the value h, the remainder of the game can be modeled by a hypergeometric distribution. Informally, B elements from a population of size $|T_v \setminus T_w|$ containing $\approx 2^{-d}$ successes are drawn without replacement. Let $X_{\delta_{d,h}}$ be a random variable indicating the number of leaves with the value h found in B draws in a sub-tree T_v such that v is in the d-th "bucket". Thus,

$$\Pr\left[\mathsf{TreeCollGame}(A, T) = 1\right] \geq \sum_{h=0}^{l-1} \sum_{d=2}^{N} \delta_{d,h} \cdot 2^{-d} \cdot \Pr\left[X_{\delta_{d,h}} \geq 1\right] \ .$$

In Section 6.2, we show that the above term is bounded by $\approx \epsilon^{1.5}/l$. □

2.5 General batch Sigma protocols

Batch Sigma protocols. In the general case, we consider batch Σ-protocols where given k instance-witness pairs $(\mathbb{x}_i, \mathbb{w}_i)$, the prover $\mathbf{P}_1$ sends a message α, the verifier $\mathbf{V}$ samples a challenge β and sends it, the prover $\mathbf{P}_2$ responds with a message γ, and the verifier $\mathbf{V}$ decides whether to accept or reject by applying a predicate to $(\mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma)$. In order to bound the rogue-soundness error of batch Σ-protocols, we make use of the special soundness property. In particular, we consider the *plus-one special soundness* which guarantees the existence of an extractor E. When it is given as input $k+1$ transcripts of an execution of a batch Sigma protocol on k instances, the extractor outputs k corresponding witnesses. More precisely, the extractor is given as input $k+1$ transcripts with a common first message and distinct pairwise challenges.

We construct an algorithm A that given as input an instance $\mathbb{x}$ invokes a malicious prover on input $\mathbb{x}$ to obtain $k+1$ transcripts, which by the plus-one special soundness allows extracting k witnesses, specifically, to output a witness for $\mathbb{x}$. Note that the algorithm needs to invoke the prover multiple times in order to achieve approximately the same probability as in the specific case of batch protocols constructed from algebraic Σ-protocols. Unfortunately, it appears that finding a good trade-off between the second-moment of the expected running time and the success probability of the algorithm is challenging in this context. As a result, in the general case, we rely on the first-moment assumption.

Similarly, we reduce the problem of finding $k+1$ accepting transcripts to a generalized version of the collision game first introduced in [Cra96]. In more detail, we construct an algorithm for the collision game and then use it in order to obtain $k+1$ accepting transcripts (with a common first message and pairwise distinct challenges), which conclude extracting a witness.

General collision game. We provide a general version of the collision game first introduced in [Cra96] and used in [ACK21, HL10], which consists of a binary matrix $H \in \{0,1\}^{R \times N}$. We generalize the collision game by an additional input, a number $k \in \mathbb{N}$. The output of the game is 1 if and only if $k+1$ entries with the value 1 in the same row have been found. An algorithm for the collision game is given as input a number $k \in \mathbb{N}$ and an oracle access to the matrix H.

Informally, the R rows correspond to the prover's randomness and the N columns correspond to the verifier's randomness. An entry of H equals 1 if and only if the corresponding transcript is accepting. Then, finding $k+1$ entries with the value 1 in the same row corresponds to finding $k+1$ accepting transcripts with a common first message and pairwise distinct challenges. Therefore, an algorithm for the collision game can be transformed into an algorithm that finds $k+1$ accepting transcripts, which as discussed above, allows extracting a witness (see Section 8.2 for a complete proof).

Lemma 3 (Informal)**.** *Let $H \in \{0,1\}^{R \times N}$ be a binary matrix and let ϵ be the fraction of 1-entries in H. Then, there exists an algorithm A with oracle access to H such that on input k the following holds:*

1. *The expected number of queries performed by A to H is at most $k+1$.*
2. *The probability that A succeeds in the game is at least ϵ.*

Proof sketch. We consider the following algorithm:

$A^H(k)$

1. Sample an entry (ρ, β) in H. If $H[\rho, \beta] = 0$, abort.
2. Sample without replacement entries in the same row ρ, until $k+1$ entries with the value 1 are found or the row has been exhausted.

Let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to H. Note that the number of 1-entries in each row affects the expected number of queries performed by A. Thus, we let ϵ_ρ be the fraction of 1-entries in row ρ. Assuming the first query to H lies in row ρ and equals 1, the remainder of the algorithm can be modeled by a negative hypergeometric distribution. Elements from a population of

size $N-1$ containing $\epsilon_\rho N - 1$ successes are drawn without replacement till k successes are counted. Thus, assuming that the first query lies in a row ρ and equals 1, the expected number of queries performed by A is $\frac{k(N-1+1)}{\epsilon_\rho N - 1 + 1} = \frac{k}{\epsilon_\rho}$. Overall,

$$\mathbb{E}\left[\mathcal{Q}_A\right] = 1 + \frac{1}{R}\sum_{1}^{R} \epsilon_\rho \cdot \frac{k}{\epsilon_\rho} = k + 1 \ .$$

As discussed in Section 2.3, in order to bound the success probability we divide the rows by the number of 1-entries in it. Formally, for every $0 \leq d \leq N$, we let δ_d be the fraction of rows with exactly d 1-entries. Note that if A's first query to H lies in a row with at least $k+1$ entries with the value 1, it succeeds in the game with probability 1. Thus,

$$\Pr\left[\mathsf{CollGame}_k(A, H) = 1\right] \geq \sum_{d=k+1}^{R} \delta_d \cdot \frac{d}{N} \ .$$

In Section 8.1 we show that the above term is bounded by $\approx \epsilon$. □

2.6 Expected time hardness framework

In this subsection, we present our framework for analyzing the expected-time hardness of cryptographic problems in generic models. Our framework allows bounding the success probability of query-algorithms in experiments that involve access to an oracle (e.g., solving discrete-logarithm in the generic group model). Here, we consider the number of queries performed by the algorithm and ignore its actual runtime.

Our overall goal is to prove statements of the form: if any algorithm that performs t queries (as a strict parameter) has success probability $\epsilon(t)$ in a particular experiment, then any algorithm A has success probability $\mathbb{E}\left[\epsilon(T_A)\right]$, where T_A is a *random variable* for the number of queries performed by A. Such a statement would allow us to derive the desired first-moment and second-moment hardness that we need for discrete-logarithm and other problems.

Perhaps surprisingly, such a general statement is *incorrect*, which we demonstrate via the multiple discrete-logarithm problem. Yun [Yun15] showed that any generic t-time algorithm given k instances of the discrete-logarithm problem solves all of them with probability at most $\epsilon(t) \leq (k \cdot t^2/p)^k$ (which is tight). However, this bound does not translate to $\mathbb{E}\left[\epsilon(T_A)\right] = k^k \cdot \mathbb{E}\left[T_A^{2k}\right]/p^k$. To illustrate this, consider the following generic algorithm A for the case where $k = 2$:

1. Perform $p^{1/4}$ distinct queries to the group generation oracle and store the query-answer list μ.
2. If there does not exist $(x, y), (x', y') \in \mu$, such that $x \neq x'$ and $y = y'$, abort.
3. Otherwise, perform another $p^{1/2}$ queries to the group generation oracle.

A careful analysis shows that the success probability of this algorithm is $\approx 1/\sqrt{p}$ and the 4-moment of the expected number of queries is $\approx p$, which does not satisfy the bound of $\epsilon \leq 4 \cdot \mathbb{E}\left[T_A^4\right]/p^2$.

This raises the question of when can we derive bounds for expected algorithms. What distinguishes the multiple discrete-logarithm (for which we have no non-trivial bounds for expected algorithms) compared to the single discrete-logarithm (for which we derive tight bounds for expected algorithms)? We define a (relatively natural) property of the experiment, called *history oblivious*, that can precisely distinguish the two cases and allows us to derive our bounds. Roughly speaking, history oblivious experiment is defined via the existence of a predicate on the sequence of query/answer pairs (the trace). When the predicate of the trace is true, then the algorithm is able to solve its task with no additional queries. When the predicate is false, the trace has a limited effect on its success probability (only the size of the trace affects the probability and not its contents).

For example, in the discrete-logarithm problem, the trace to the generic group would be true if it contains a collision. When the predicate is true, one can easily deduce a solution. Otherwise, the trace gives almost no helpful information to the algorithm except for specific elements which are not the discrete-logarithm. That is, in this case, the advantage only depends on the size of the trace. Any two traces of the same size

for which the predicate is false yield equal success probability for the algorithm. Observe that this is not the case for multiple discrete-logarithm. Here, we have three types of interesting traces (rather than two). A trace can contain no collisions, or a single collision (from which one can deduce one discrete-logarithm but not the other), or two collisions (from which one can derive both discrete-logarithms). The predicate in this case would identify a trace with two collisions. Thus, two traces of the same size, one from the first type and one from the second type would have drastic different effect on the success probability, as in the latter it needs to solve only a single discrete-logarithm.

In summary, for any history oblivious experiment we show that:

$$\Pr[\textbf{strict} \text{ algorithms succeeds}] \leq \epsilon(t) \implies \Pr[\textbf{expected-time} \text{ algorithms succeeds}] \leq \mathbb{E}\left[\epsilon(t)\right] \quad .$$

We formalize the above statement in Theorem 9.5. This allows us to prove first and second-moment hardness of discrete-logarithm Equations (1) and (2), which are the basis for our results. It also allows us to derive our bounds for the Micali SNARK construction given in Theorem 5. Our framework is inspired by the work of Jaeger and Tessaro [JT20], however, their tools do not allow us to prove the second-moment hardness assumptions in generic models. Furthermore, our approach is arguably simpler to use and provides tighter security bounds even for first-moment assumptions. We show that our framework recovers the bounds of [JT20] in Corollary 9.6.

3 Preliminaries

For any $n \in \mathbb{N}$, we denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$.

3.1 High-moment hardness

A relation $\mathcal{R}$ is a set $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$, for sets $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{W} = \{\mathcal{W}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$. The corresponding language $\mathcal{L}(\mathcal{R}_\lambda)$ is the set of public parameters pp and instances $\mathbb{x}$ for which there exists a witness $\mathbb{w}$ such that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$.

We consider distributions $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ over the relation where each $\mathcal{D}_\lambda$ produces $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$. We note by $\mathcal{D}_\lambda(\mathsf{pp})$ the distribution that produces $(\mathbb{x}, \mathbb{w})$ such that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$.

For any such distribution $\mathcal{D}_\lambda(\mathsf{pp})$ and an algorithm A, we denote by $T_{\mathcal{A},\mathcal{D}_\lambda}$ the random variable indicating the running time of A on input $\mathbb{x}$ where $(\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}_\lambda(\mathsf{pp})$.

Definition 3.1 (First-moment hard relation)**.** *Let $\Delta = \Delta(\lambda), \omega = \omega(\lambda)$ be functions of the security parameter, and let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a relation where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$. Let Setup be a setup algorithm that on input 1^λ, outputs $\mathsf{pp} \in \mathcal{P}_\lambda$. We say that $\mathcal{R}$ is first-moment hard (with respect to a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ and a setup algorithm Setup) if for every algorithm A and for every $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[(\mathsf{pp}, \mathbb{x}, \tilde{\mathbb{w}}) \in \mathcal{R}_\lambda \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}} \leftarrow A(\mathsf{pp}, \mathbb{x}) \end{array}\right] \leq \frac{\Delta \cdot \mathbb{E}\left[T_{A,\mathcal{D}_\lambda}\right]}{|\mathcal{W}_\lambda|^\omega} \ .$$

Definition 3.2 (Second-moment hard relation)**.** *Let $\Delta = \Delta(\lambda), \omega = \omega(\lambda)$ be functions of the security parameter, and let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a relation where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$. Let Setup be a setup algorithm that on input 1^λ, outputs $\mathsf{pp} \in \mathcal{P}_\lambda$. We say that $\mathcal{R}$ is second-moment hard (with respect to a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ and a setup algorithm Setup) if for every algorithm A and for every $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[(\mathsf{pp}, \mathbb{x}, \tilde{\mathbb{w}}) \in \mathcal{R}_\lambda \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}} \leftarrow A(\mathsf{pp}, \mathbb{x}) \end{array}\right] \leq \frac{\Delta \cdot \mathbb{E}\left[T^2_{A,\mathcal{D}_\lambda}\right]}{|\mathcal{W}_\lambda|^\omega} \ .$$

3.2 Sigma protocols

Definition 3.3 (Σ-Protocol)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$. A Σ-protocol Π for relation $\mathcal{R}$ is a 5-tuple $(\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ where Setup and $\boldsymbol{P}_1$ are probabilistic polynomial-time algorithms, $\boldsymbol{P}_2$ and $\boldsymbol{V}$ are deterministic polynomial-time algorithms, and $\mathcal{C} = \{\mathcal{C}_{\mathsf{pp}}\}_{\mathsf{pp}\in\mathcal{P}}$ is an ensemble of efficiently sampleable sets. The protocol Π is defined as follows:*

1. *The algorithm $\mathsf{Setup}(1^\lambda)$ produces public parameters pp.*
2. *The algorithm $\boldsymbol{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ produces a message α and a state st.*
3. *A challenge β is sampled uniformly at random from the challenge set $\mathcal{C}_{\mathsf{pp}}$.*
4. *The algorithm $\boldsymbol{P}_2(\mathsf{st}, \beta)$ produces a message γ.*
5. *The algorithm $\boldsymbol{V}(\mathsf{pp}, \mathbb{x}, \alpha, \beta, \gamma)$ determines the output of the protocol by outputting 0 or 1.*

We require that for every $\lambda \in \mathbb{N}$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ it holds that

$$\Pr\left[\boldsymbol{V}(\mathsf{pp}, \mathbb{x}, \alpha, \beta, \gamma) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\alpha, \mathsf{st}) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \\ \beta \leftarrow \mathcal{C}_{\mathsf{pp}} \\ \gamma \leftarrow \boldsymbol{P}_2(\mathsf{st}, \beta) \end{array}\right] = 1 \ .$$

Definition 3.4 (Special soundness)**.** *Let* $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a* Σ*-protocol for a relation* $\mathcal{R}$*, and let* $t = t(\lambda)$ *be a function of the security parameter* $\lambda \in \mathbb{N}$*. Then,* Π *has* t*-time special soundness if there exists a deterministic* t*-time algorithm* E *that on any public parameters* $\mathsf{pp} \in \mathcal{P}$*, any input statement* $\mathbb{x} \in \mathcal{X}_\lambda$ *and any two accepting transcripts with a common first message and distinct challenges, outputs a witness* $\mathbb{w}$ *such that* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$*.*

Definition 3.5 (Zero knowledge Σ-protocol)**.** *Let* $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a* Σ*-protocol for a relation* $\mathcal{R}$*, and let* $t = t(\lambda)$ *be a function of the security parameter* $\lambda \in \mathbb{N}$*. Then,* Π *is* t*-time zero-knowledge if there exists a probabilistic* t*-time algorithm* Sim *such that for every* $\lambda \in \mathbb{N}$ *and public parameters-instance-witness tuple* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ *the distributions*

$$\left\{ (\mathsf{pp}, \mathbb{x}, \alpha, \beta, \gamma) \;\middle|\; \begin{array}{l} (\alpha, \mathsf{st}) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \\ \beta \leftarrow \mathcal{C}_{\mathsf{pp}} \\ \gamma \leftarrow \boldsymbol{P}_2(\mathsf{st}, \beta) \end{array} \right\} \qquad \textit{and} \qquad \{\mathsf{Sim}(\mathsf{pp}, \mathbb{x})\}$$

are identical.

3.3 Batch Sigma protocols

Definition 3.6 (Batch Σ-protocol)**.** *Let* $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a relation, where* $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ *for any* $\lambda \in \mathbb{N}$ *and let* $\boldsymbol{K} \in \mathbb{N}$ *be a bound on the number of instances. A batch* Σ*-protocol* Π *for relation* $\mathcal{R}$ *is a* 5*-tuple* $(\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *where* Setup *and* $\boldsymbol{P}_1$ *are probabilistic polynomial-time algorithms,* $\boldsymbol{P}_2$ *and* $\boldsymbol{V}$ *are deterministic polynomial-time algorithms, and* $\mathcal{C} = \{\mathcal{C}_{\mathsf{pp}}\}_{\mathsf{pp} \in \mathcal{P}}$ *is an ensemble of efficiently sampleable sets. For any* $k \leq \boldsymbol{K}$*, the protocol* Π *is defined as follows:*

1. *The algorithm* $\boldsymbol{P}_1(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k))$ *produces a message* α *and a state* st*.*
2. *A challenge* β *is sampled uniformly at random from the challenge set* $\mathcal{C}_{\mathsf{pp}}$*.*
3. *The algorithm* $\boldsymbol{P}_2(\mathsf{st}, \beta)$ *produces a message* γ*.*
4. *The algorithm* $\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma)$ *determines the output of the protocol by outputting* 0 *or* 1*.*

We have the following two requirements:

1. ***Completeness:*** *For every* $\lambda, k \in \mathbb{N}$ *such that* $k \leq \boldsymbol{K}$*, for any* $(\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k) \in \mathcal{R}_\lambda$ *it holds that*

$$\Pr\left[\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\alpha, \mathsf{st}) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)) \\ \beta \leftarrow \mathcal{C}_{\mathsf{pp}} \\ \gamma \leftarrow \boldsymbol{P}_2(\mathsf{st}, \beta) \end{array} \right] = 1 \ .$$

2. ***Soundness:*** Π *has* ϵ *(adaptive) soundness error if for every* $\lambda, k \in \mathbb{N}$ *such that* $k \leq \boldsymbol{K}$*, it holds that for any malicious prover* $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2)$*:*

$$\Pr\left[\begin{array}{l} \exists i \in [k],\ s.t.\ \mathbb{x}_i \in \mathcal{X}_\lambda \setminus \mathcal{L}(\mathcal{R}) \\ \boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma) = 1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \mathsf{st}) \leftarrow \tilde{\boldsymbol{P}}_1(\mathsf{pp}) \\ \beta \leftarrow \mathcal{C}_{\mathsf{pp}} \\ \gamma \leftarrow \tilde{\boldsymbol{P}}_2(\mathsf{st}, \beta) \end{array} \right] \leq \epsilon(\lambda, \boldsymbol{K}) \ .$$

Definition 3.7 (Plus-one special soundness)**.** *Let* $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a batch* Σ*-protocol for a relation* $\mathcal{R}$ *with a bound* $\boldsymbol{K}$ *on the number of instances, and let* $t = t(\lambda, \boldsymbol{K})$ *be a function of* $\boldsymbol{K}$ *and the security parameter* $\lambda \in \mathbb{N}$*. Then,* Π *has* t*-time plus-one special soundness if there exists a deterministic* t*-time algorithm* E *that for every* $\lambda \in \mathbb{N}$ *and* $k \leq \boldsymbol{K}$*, on any public parameters* pp*, any* k *inputs statements* $\mathbb{x}_1, \ldots, \mathbb{x}_k \in \mathcal{X}_\lambda$ *and any* $k + 1$ *accepting transcripts with a common first message and pairwise distinct challenges, outputs* k *witnesses* $\mathbb{w}_1, \ldots, \mathbb{w}_k$ *such that for every* $i \in [k]$ *it holds that* $(\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}_\lambda$*.*

Definition 3.8 (Zero knowledge batch Σ-protocol)**.** *Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be a batch Σ-protocol for a relation $\mathcal{R}$ with a bound $\boldsymbol{K}$ on the number of instances, and let $t = t(\lambda, \boldsymbol{K})$ be a function of $\boldsymbol{K}$ and the security parameter $\lambda \in \mathbb{N}$. Then, Π is t-time zero-knowledge if there exists a probabilistic t-time algorithm Sim such that for any $k \leq \boldsymbol{K}$, for every $\lambda \in \mathbb{N}$ and $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathsf{pp}, \mathbb{x}_k, \mathbb{w}_k) \in \mathcal{R}_\lambda$ the distributions*

$$\left\{ (\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma) \;\middle|\; \begin{array}{l} (\alpha, \mathsf{st}) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)) \\ \beta \leftarrow \mathcal{C}_{k,\lambda} \\ \gamma \leftarrow \boldsymbol{P}_2(\mathsf{st}, \beta) \end{array} \right\} \quad \textit{and} \quad \{\mathsf{Sim}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k)\}$$

are identical.

3.4 Non-interactive arguments

Definition 3.9 (Non-interactive argument)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$. A non-interactive argument (NARG) in the ROM for relation $\mathcal{R}$ is a 3-tuple $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ where Setup, $\mathcal{P}$, and $\mathcal{V}$ are oracle algorithms such that the following holds:*

1. ***Completeness:*** *For every security parameter $\lambda \in \mathbb{N}$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ it holds that*

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

2. ***Soundness:*** *NARG has ϵ (adaptive) soundness error if for every $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$ and for any t-time malicious prover $\tilde{\mathcal{P}}$ that performs at most q queries to f it holds that:*

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \mathbb{x} \in \mathcal{X}_\lambda \setminus \mathcal{L}(\mathcal{R}) \\ \mathcal{V}^f(\mathsf{pp}, \mathbb{x}, \tilde{\pi}) = 1 \end{array} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda) \\ (\mathbb{x}, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^f(\mathsf{pp}) \end{array} \right] \leq \epsilon(\lambda, t, \mathsf{q}, n) \ .$$

Definition 3.10 (Zero knowledge)**.** *Let $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a non-interactive argument for a relation $\mathcal{R}$, and let $t = t(\lambda)$ be a function of the security parameter $\lambda \in \mathbb{N}$. Then, NARG has adaptive zero-knowledge if there exists a probabilistic t-time algorithm Sim, such that for every $\lambda \in \mathbb{N}$, query bound $\mathsf{q} \in \mathbb{N}$, q-query admissible algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and instance bound $n \in \mathbb{N}$, the following two distributions are identical:*

$$\left\{ b \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda) \\ (\mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow A_1^f(\mathsf{pp}) \\ \pi \leftarrow \mathcal{P}^f(\mathbb{x}, \mathbb{w}) \\ b \leftarrow \mathcal{A}_2^f(\mathsf{st}, \pi) \end{array} \right\} \qquad \textit{and} \qquad \left\{ b' \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda) \\ (\mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow A_1^f(\mathsf{pp}) \\ (\pi, \mu) \leftarrow \mathsf{Sim}^f(\mathbb{x}) \\ b' \leftarrow \mathcal{A}_2^{f[\mu]}(\mathsf{st}, \pi) \end{array} \right\} \ ,$$

where $f[\mu]$ denotes the fact that f is modified to be consistent with the query-answer list μ. Above, $\mathcal{A}$ is admissible if on input pp it always outputs $(\mathbb{x}, \mathbb{w})$ such that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ and $|\mathbb{x}| \leq n$.

3.5 Non-interactive batch arguments

Definition 3.11 (Non-interactive batch argument)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$ and let $\boldsymbol{K}$ be a bound on the number of instances. A non-interactive batch argument in the ROM for relation $\mathcal{R}$ is a 3-tuple $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ where Setup is a probabilistic polynomial time algorithm, and $\mathcal{P}$ and $\mathcal{V}$ are oracle algorithms such that the following holds:*

1. ***Completeness:*** *For every security parameter $\lambda, k \in \mathbb{N}$ such that $k \leq \boldsymbol{K}$, and for any k pairs $(\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k) \in \mathcal{R}_\lambda$ it holds that*

$$\Pr\left[\boldsymbol{V}^f(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \pi) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \boldsymbol{K}) \\ \pi \leftarrow \mathcal{P}^f(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)) \end{array} \right] = 1 \ .$$

2. ***Soundness:*** NARG *has* ϵ *(adaptive) soundness error if for every* $\lambda, k \in \mathbb{N}$ *such that* $k \leq \boldsymbol{K}$*, instance size bound* $n \in \mathbb{N}$*, and for any* t*-time malicious prover* $\tilde{\mathcal{P}}$ *that performs at most* q *queries to* f *it holds that:*

$$\Pr\left[\begin{array}{l|l} \forall i \in [k] : |\mathbb{x}_i| \leq n & f \leftarrow \mathcal{U}(\lambda) \\ \exists j \in [k] : \mathbb{x}_j \in \mathcal{X}_\lambda \setminus \mathcal{L}(\mathcal{R}) & \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \boldsymbol{K}) \\ \mathcal{V}^f(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \tilde{\pi}) = 1 & (\mathbb{x}_1, \ldots, \mathbb{x}_k, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^f(\mathsf{pp}) \end{array}\right] \leq \epsilon(\lambda, t, \mathsf{q}, n) \ .$$

Definition 3.12 (Zero knowledge)**.** *Let* $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *be a non-interactive batch argument for a relation* $\mathcal{R}$ *with a bound* $\boldsymbol{K}$ *on the number of instances, and let* $t = t(\lambda, \boldsymbol{K})$ *be a function of the security parameter* $\lambda \in \mathbb{N}$*. Then,* NARG *has adaptive zero-knowledge if there exists a probabilistic* t*-time algorithm* Sim*, such that for every* $\lambda, k \in \mathbb{N}$ *such that* $k \leq \boldsymbol{K}$*, query bound* $\mathsf{q} \in \mathbb{N}$*,* q*-query admissible algorithm* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and instance size bound* $n \in \mathbb{N}$*, the following two distributions are identical:*

$$\left\{ b \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \boldsymbol{K}) \\ ((\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k), \mathsf{st}) \leftarrow A_1^f(\mathsf{pp}) \\ \pi \leftarrow \mathcal{P}^f(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)) \\ b \leftarrow \mathcal{A}_2^f(\mathsf{st}, \pi) \end{array} \right\}$$

and,

$$\left\{ b' \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \boldsymbol{K}) \\ ((\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k), \mathsf{st}) \leftarrow A_1^f(\mathsf{pp}) \\ (\pi, \mu) \leftarrow \mathsf{Sim}^f(\mathbb{x}_1, \ldots, \mathbb{x}_k) \\ b' \leftarrow \mathcal{A}_2^{f[\mu]}(\mathsf{st}, \pi) \end{array} \right\} \ ,$$

where $f[\mu]$ *denotes the fact that* f *is modified to be consistent with the query-answer list* μ*. Above,* $\mathcal{A}$ *is admissible if on input* pp *it always outputs* $(\mathbb{x}, \mathbb{w})$ *such that* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ *and* $|\mathbb{x}| \leq n$*.*

3.6 Probabilities

Definition 3.13 (Hypergeometric distribution)**.** *The hypergeometric distribution describes the probability of* k *"successes", in* n *draws without replacement, from a population of size* N *of which* K *are defined as "successes". If a random variable follows the hypergeometric distribution it holds that*

$$\Pr[X = k] = \frac{\binom{K}{k} \cdot \binom{N-K}{n-k}}{\binom{N}{n}} \ .$$

Definition 3.14 (Negative hypergeometric distribution)**.** *The negative hypergeometric distribution describes the following distribution: Given* N *elements, of which* K *are defined as "successes" and the rest are "failures", elements are drawn one after another, without replacement. Let* X *be the number of draws until* k *successes are drawn. Thus,*

$$\Pr[X = x] = \binom{x-1}{k-1} \cdot \frac{\binom{N-x}{K-k}}{\binom{N}{K}} \ ,$$

and,

$$\mathbb{E}[X] = \frac{k(N+1)}{K+1} \ .$$

4 Rogue-instance security

In this section, we give our definition of rogue-instance security notion for batch protocols and non-interactive batch arguments, which is inspired by the rogue-key security notion for multi-signatures.

4.1 Batch Sigma protocols

In a batch Σ-protocol, we are given k instance-witness pairs $(\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)$. The standard adaptive soundness requirement considers the case where a malicious prover wishes to convince the verifier on k instances of its choice. However, we consider batch Σ-protocols with rogue-instance security, where one instance $\mathbb{x}_1$ is sampled according to a given hard distribution, and the rest of the instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ are chosen adaptively as a function of $\mathbb{x}_1$. Formally,

Definition 4.1 (Rogue soundness)**.** *Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be a batch Σ-protocol for a relation $\mathcal{R}$ with a bound $\boldsymbol{K}$ on the number of instances. Then, Π has $(t, \epsilon_{\mathcal{D}})$-rogue soundness (with respect to a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and the setup algorithm Setup) if for every $\lambda, k \in \mathbb{N}$ such that $k \leq \boldsymbol{K}$ and for any t-time malicious prover $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2)$:*

$$\Pr\left[\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \beta, \gamma) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ ((\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k), \alpha, \mathsf{st}) \leftarrow \tilde{\boldsymbol{P}}_1(\mathsf{pp}, \mathbb{x}_1) \\ \beta \leftarrow \mathcal{C}_{\mathsf{pp}} \\ \gamma \leftarrow \tilde{\boldsymbol{P}}_2(\mathsf{st}, \beta) \end{array}\right] \leq \epsilon_{\mathcal{D}}(\lambda, t, \boldsymbol{K}) \ .$$

4.2 Non-interactive batch arguments

Similarly, in a non-interactive batch argument, we are given k instance-witness pairs $(\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k)$. We consider the rogue-instance security, where an instance $\mathbb{x}_1$ is sampled according to a given hard distribution, and the rest of the instances $\mathbb{x}_2, \ldots, \mathbb{x}_k$ are chosen adaptively by the prover while having oracle access to a random oracle. Formally, we define the rogue-instance security of non-interactive batch arguments as follows:

Definition 4.2 (Rogue soundness)**.** *Let $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a non-interactive batch argument for a relation $\mathcal{R}$ with a bound $\boldsymbol{K}$ on the number of instances. Then, NARG has $(t, \mathsf{q}, \epsilon_{\mathcal{D}})$-rogue soundness (with respect to a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and the setup algorithm Setup) if for every $\lambda, k \in \mathbb{N}$ such that $k \leq \boldsymbol{K}$ and for any t-time malicious prover $\tilde{\mathcal{P}}$ that performs at most q queries to f it holds that:*

$$\Pr\left[\mathcal{V}^f(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \tilde{\pi}) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Setup}^f(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ (\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^f(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \leq \epsilon_{\mathcal{D}}(\lambda, t, \mathsf{q}, k) \ .$$

5 Batching algebraic Sigma protocols

In this section, we define algebraic Σ-protocols and construct their batch version. Then, we bound the rogue-soundness error of such batch Σ-protocols using the second-moment assumption (Definition 3.2).

In Section 5.1 we define algebraic one-way functions and construct batch Σ-protocols from algebraic Σ-protocols. Then, in Section 5.2 we generalize the "collision game" presented in [ACK21, HL10, Cra96] for multiple instances while referring to the second-moment of the expected running time. Finally, in Section 5.3 we prove the rogue-instance security of batch Σ-protocols constructed from algebraic Σ-protocols.

5.1 Algebraic Sigma protocols

In this section, we refer to Σ-protocols that have a specific structure we call *algebraic Σ-protocols* and then, we define their batch analog.

Our definition of algebraic Σ-protocols relies on algebraic one-way function, presented in [CFG+15, CD98].

Definition 5.1 (Algebraic one-way function)**.** *A family of m-variate one-way functions consists of two algorithms $(\mathsf{Setup}, \mathsf{F})$ that work as follows. On input 1^λ, the algorithm $\mathsf{Setup}(1^\lambda)$ produces public parameters. Any such public parameters pp, determines the function $\mathsf{F}_{\mathsf{pp}} \colon \mathcal{A}_{\mathsf{pp}}^m \to \mathcal{B}_{\mathsf{pp}}$ such that for every $x \in \mathcal{A}_{\mathsf{pp}}^m$, it is efficient to compute $\mathsf{F}_{\mathsf{pp}}(x)$. A family of one-way functions is algebraic if for every $\lambda \in \mathbb{N}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ the following holds:*

- ***Algebraic:*** *The sets $\mathcal{A}_{\mathsf{pp}}, \mathcal{B}_{\mathsf{pp}}$ are abelian cyclic groups with operators $(+)$, and $(\cdot)$, respectively.*
- ***Homomorphic****: For any input $x, x' \in \mathcal{A}_{\mathsf{pp}}^m$ it holds that $\mathsf{F}(x + x') = \mathsf{F}(x) \cdot \mathsf{F}(x')$.*

We now define the notion of algebraic Σ-protocols, which is a generalization of the *preimage protocol* presented in [CD98].

Definition 5.2 (Algebraic Σ-protocol)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$. A Σ-protocol $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ for relation $\mathcal{R}$ is algebraic if there exists m-variate algebraic one-way function $(\mathsf{Setup}, \mathsf{F})$ such that for every pp produced by $\mathsf{Setup}(1^\lambda)$ the following holds:*

- *For every $\mathbb{x}, \mathbb{w}$ it holds that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$ if and only if $\mathsf{F}_{\mathsf{pp}}(\mathbb{w}) = \mathbb{x}$.*
- *The challenge space $\mathcal{C}_{\mathsf{pp}} \subseteq \mathbb{Z}_p$ where p is the order of $\mathcal{A}_{\mathsf{pp}}$.*
- *The protocol Π is defined as follows:*
  1. *The algorithm $\boldsymbol{P}_1(\mathbb{x}, \mathbb{w})$ produces a message $\alpha = \mathsf{F}(r)$ for some $r \in \mathcal{A}_{\mathsf{pp}}$ and a state st.*
  2. *A challenge β is sampled uniformly at random from the challenge set $\mathcal{C}_{\mathsf{pp}}$.*
  3. *The algorithm $\boldsymbol{P}_2(\mathsf{st}, \beta)$ produces a message $\gamma = r + \beta \cdot \mathbb{w}$.*
  4. *The algorithm $\boldsymbol{V}(\mathbb{x}, \alpha, \beta, \gamma)$ determines the output of the protocol by checking whether $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \mathbb{x}^\beta$.*

Note that the setup algorithm of the function is the setup algorithm of the protocol. In fact, the prover holds a public parameters-instance-witness tuple such that $\mathbb{x} = \mathsf{F}_{\mathsf{pp}}(\mathbb{w})$. Thus, the prover convinces the verifier that it knows the preimage of $\mathbb{x}$. Note that the verifier's computation can be performed using exponentiation by squaring, however there may exist more efficient algorithms.

Next, we construct a batch version of any algebraic Σ-protocol as follows.

Construction 5.3 (Batch Σ-protocol)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$ and let $\boldsymbol{K} \in \mathbb{N}$ be a bound on the number of instances. Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be an algebraic Σ-protocol with an agebraic one-way function $(\mathsf{Setup}, \mathsf{F})$. We define $\Pi^* = (\mathsf{Setup}^*, \mathbf{P}_\mathbf{1}^*, \mathbf{P}_\mathbf{2}^*, \mathbf{V}^*, \mathcal{C})$ to be a batch Σ-protocol for relation $\mathcal{R}$ as follows. The algorithms Setup^* and $\mathbf{P}_\mathbf{1}^*$ are probabilistic polynomial-time algorithms, $\mathbf{P}_\mathbf{2}^*$ and $\mathbf{V}^*$ are deterministic polynomial-time algorithms, and $\mathcal{C} = \{\mathcal{C}_{\mathsf{pp}}\}_{\mathsf{pp} \in \mathcal{P}}$ is an ensemble of efficiently sampleable sets. For every $k \leq \boldsymbol{K}$ the protocol is defined as follows:*

1. *The algorithm* $\mathsf{Setup}^*(1^\lambda, \boldsymbol{K})$ *is the same algorithm as* $\mathsf{Setup}(1^\lambda)$.
2. *The algorithm* $\mathbf{P}_\mathbf{1}^*(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k))$ *invokes* $(R_i, \mathsf{st}_i) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i)$ *for every* $i \in [k]$ *and produces a message* $\alpha = \Pi_{i=1}^k R_i$ *and a state* $\mathsf{st} = (\mathsf{st}_1 \| \ldots \| \mathsf{st}_k)$.
3. k *different challenges* $\beta_1, \ldots, \beta_k$ *are sampled uniformly at random from the challenge set* $\mathcal{C}_{\mathsf{pp}}$.
4. *The algorithm* $\mathbf{P}_\mathbf{2}^*(\mathsf{st}, \beta_1, \ldots, \beta_k)$ *parses* $\mathsf{st} = (\mathsf{st}_1 \| \ldots \| \mathsf{st}_k)$, *invokes* $\gamma_i \leftarrow \boldsymbol{P}_2(\mathsf{st}_i, \beta_i)$ *and produces a message* $\gamma = \sum_{i=1}^k \gamma_i$.
5. *The algorithm* $\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta, \gamma)$ *determines the output of the protocol checking whether* $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \Pi_{i=1}^k \mathbb{x}_i^{\beta_i}$.

Note that the completeness of the protocol above follows from the homomorphic property of F and that the prover-to-verifier communication is two-group elements. The verifier sends k elements, but since they are all uniformly random strings, they can be easily compressed to a single group element using any pseudo-random generator (e.g., using a random oracle).

Definition 5.4 (Local special soundness)**.** *Let* $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be an algebraic* Σ*-protocol for a relation* $\mathcal{R}$ *and let* Π^* *be the batch* Σ*-protocol defined in Construction 5.3 with a bound* $\boldsymbol{K}$ *on the number of instances. Then,* Π^* *has local special soundness if there exists a deterministic polynomial time algorithm* E *that for every* $\lambda \in \mathbb{N}$ *and* $k \leq \boldsymbol{K}$*, given public parameters* pp*, any* k *inputs statements* $\mathbb{x}_1, \ldots, \mathbb{x}_k \in \mathcal{X}_\lambda$ *and any pair of accepting transcripts* $(\alpha, \beta_1, \ldots, \beta_k, \gamma), (\alpha, \beta_1', \ldots, \beta_k', \gamma')$ *such that there exists only one index* j *on which* $\beta_j \neq \beta_j'$*, outputs a witness* $\mathbb{w}_j$ *such that* $(\mathbb{x}_j, \mathbb{w}_j) \in \mathcal{R}_\lambda$.

We now show that every batch Σ-protocol defined in Construction 5.3 has local special soundness.

Claim 5.5. *Let* $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be an algebraic* Σ*-protocol for a relation* $\mathcal{R}$ *and let* Π^* *be the batch* Σ*-protocol constructed from* Π *as defined in Construction 5.3 with a bound* $\boldsymbol{K}$ *on the number of instances. Then,* Π^* *has local special soundness.*

Proof. Consider the algorithm E which takes as input public parameters pp, instances $\mathbb{x}_1, \ldots, \mathbb{x}_k$ and a pair of accepting transcripts $(\alpha, \beta_1, \ldots, \beta_k, \gamma), (\alpha, \beta_1', \ldots, \beta_k', \gamma')$ such that there exists only one index j on which $\beta_j \neq \beta_j'$, defined as follows:

1. Let i^* be the index on which $\beta_{i^*} \neq \beta_{i^*}'$.
2. Output $(\gamma - \gamma')/(\beta_{i^*} - \beta_{i^*}')$ on the group $\mathbb{Z}_p$ where p is the order of $\mathcal{A}_{\mathsf{pp}}$.

Observe that since the two transcripts are accepting it holds that

$$\mathsf{F}_{\mathsf{pp}}(\gamma) = \alpha \cdot \Pi_{i=1}^k \mathbb{x}_i^{\beta_i} \qquad \text{and} \qquad \mathsf{F}_{\mathsf{pp}}(\gamma') = \alpha \cdot \Pi_{i=1}^k \mathbb{x}_i^{\beta_i'} \ .$$

Since $\beta_i = \beta_i'$ for every $i \neq i^*$, it holds that

$$\mathbb{x}_{i^*}^{\beta_{i^*}} \cdot \mathsf{F}_{\mathsf{pp}}(\gamma') = \mathbb{x}_{i^*}^{\beta_{i^*}'} \cdot \mathsf{F}_{\mathsf{pp}}(\gamma) \ .$$

Note that $\mathbb{x}_{i^*} = \mathsf{F}_{\mathsf{pp}}(\mathbb{w}_{i^*})$, therefore, by the homomorphic property, it holds that

$$\mathsf{F}_{\mathsf{pp}}((\beta_{i^*} - \beta_{i^*}')\mathbb{w}_{i^*}) = \mathsf{F}_{\mathsf{pp}}(\gamma - \gamma') \ .$$

Thus, $(\gamma - \gamma')/(\beta_{i^*} - \beta_{i^*}')$ is a preimage of $\mathbb{x}_{i^*}$, i.e, a valid witness for $\mathbb{x}_{i^*}$. The extractor E performs only three group operations, therefore, Π^* has local special soundness. □

In Section 5.3, we show a concrete bound on the rogue soundness error of batch Σ-protocols defined in Construction 5.3. Formally, we prove the following.

Theorem 5.6. *Let* $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda, \boldsymbol{K}), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K})$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$*. Let* Π *be an algebraic* Σ*-protocol for a relation* $\mathcal{R}$ *and let* $\Pi^* = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be the batch* Σ*-protocol constructed from* Π *as defined*

in Construction 5.3. If $\mathcal{R}$ is second-moment hard with respect to a distribution $\mathcal{D}$ and the setup algorithm Setup*, then Π^* has $(t_{\tilde{\boldsymbol{P}}}, \epsilon)$-rogue soundness error such that*

$$\epsilon_{\mathcal{D}}(\lambda, t_{\tilde{\boldsymbol{P}}}, t_{\boldsymbol{V}}, t_W, \boldsymbol{K}) \leq \left(\frac{\Delta \cdot 32 \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)^2}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3} + \frac{4}{|\mathcal{C}_{\mathsf{pp}}|} \ ,$$

where $t_{\boldsymbol{V}}$ denotes the running time of the verifier $\boldsymbol{V}$ and t_W denotes the running time of the witness extractor.

5.2 The collision game

Similar to the collision game presented in [ACK21, HL10, Cra96], we consider a binary matrix $H \in \{0,1\}^{R \times N}$. However, instead of looking for two 1-entries in the same row, the generalized algorithm A is given as input a number $k \in \mathbb{N}$ and oracle access to the matrix and its goal is to find $k+1$ entries with the value 1 in the same row in H. Formally, the game is constructed as follows:

$\mathsf{CollGame}_k(A, H)$

1. The algorithm $A(k)$ is given oracle access to H and outputs ρ and $\beta_1, \ldots, \beta_{k+1}$.
2. The output of the game is 1 if and only if $H[\rho, \beta_1] = \ldots = H[\rho, \beta_{k+1}] = 1$ and $\beta_1, \ldots, \beta_{k+1}$ are distinct.

In particular, in this section, we refer to the collision game when $k = 1$. We construct an algorithm that finds two 1-entries in the same row in H with probability at least $\approx \epsilon^{3/2}$ and performs ≈ 2 queries to H where ϵ is the fraction of 1-entries in H. Formally, we prove the following.

Lemma 5.7. *Let $H \in \{0,1\}^{R \times N}$ be a binary matrix and let ϵ be the fraction of 1-entries in H. Let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to H. Then, there exists an algorithm A with oracle access to H such that on input $k = 1$ the following holds:*

1. $\mathbb{E}[\mathcal{Q}_A] \leq 2$.
2. $\mathbb{E}\left[\mathcal{Q}_A^2\right] \leq 4$.
3. *Either* $\epsilon < \frac{4}{N}$ *or* $\Pr[\mathsf{CollGame}(A, H) = 1] \geq \frac{\epsilon^{1.5}}{8}$.

Proof. Let $B = \left\lceil \frac{1}{\sqrt{\epsilon}} - 1 \right\rceil$ and consider the following algorithm A:

$A^H(1)$

1. Sample $\rho \leftarrow R$ and $\beta \leftarrow N$. If $H[\rho, \beta] = 0$ abort.
2. Let $S = \emptyset$. For every $i \in [B]$, sample $\beta_i \leftarrow N \setminus S$ and set $S = S \cup \{\beta_i\}$. If for every $i \in [B]$ it holds that $H[\rho, \beta_i] = 0$, abort.
3. Choose uniformly at random an index i for which $H[\rho, \beta_i] = 1$.
4. Return ρ, β and β_i.

We now prove each claim separately.

Claim 5.8. *It holds that $\mathbb{E}[\mathcal{Q}_A] \leq 2$.*

Proof. By the description of A, it performs a single query to H, and then only with probability ϵ it performs B queries. Thus, we can bound the expectation by

$$\mathbb{E}[\mathcal{Q}_A] = 1 + \epsilon \cdot B \leq 1 + \frac{1}{\sqrt{\epsilon}} \cdot \epsilon \leq 2 \ .$$

□

Claim 5.9. *It holds that* $\mathbb{E}\left[\mathcal{Q}_A^2\right] \leq 4$.

Proof. By the description of A, with probability $1-\epsilon$, it performs a single query, and with probability ϵ it performs $(1+B)$ queries. Thus, we can bound the expectation squared by

$$\begin{aligned}\mathbb{E}\left[\mathcal{Q}_A^2\right] &= (1-\epsilon)\cdot 1^2 + \epsilon\cdot(1+B)^2 \\ &= 1-\epsilon+\epsilon(1+2B+B^2) \\ &= 1+2\epsilon B+\epsilon B^2 \\ &\leq 1+2\sqrt{\epsilon}+1 \\ &\leq 4 \ .\end{aligned}$$

□

Claim 5.10 (Success probability)**.** *Either* $\epsilon < \frac{4}{N}$ *or* $\Pr[\mathsf{CollGame}(A,H)=1] \geq \frac{\epsilon^{1.5}}{8}$.

In order to bound A's success probability, we first show a lower bound on the probability that A does not abort in Item 2.

Claim 5.11. *Let* X_d *be a random variable indicating the number of* 1*-entries found in* B *draws in a row with exactly* d 1*-entries. For every* $d>1$*, it holds that* $\Pr[X_d \geq 1] \geq \min\{0.5, \frac{d\cdot B}{2N}\}$.

Proof. The random variable X_d can be modeled by a hypergeometric distribution (see Definition 3.13). The population is of size $N-1$, the number of successes is $d-1$ and the number of draws is B. Thus,

$$\begin{aligned}&\Pr[X_d \geq 1] \\ &= 1-\Pr[X_d=0] \\ &= \left(1-\frac{\binom{d-1}{0}\cdot\binom{N-1-(d-1)}{B-0}}{\binom{N-1}{B}}\right) \qquad (3) \\ &= \left(1-\frac{(N-d)!\cdot(N-1-B)!}{(N-1)!\cdot(N-d-B)!}\right) \\ &= \left(1-\frac{(N-d)(N-d-1)\cdot\ldots\cdot(N-d-B+1)}{(N-1)(N-2)\cdot\ldots\cdot(N-1-B+1)}\right) \ ,\end{aligned}$$

where Equation (3) follows from the hypergeometric distribution. Note that, for every $i \geq 1$ it holds that

$$\frac{N-d-i+1}{N-i} = 1-\frac{d-1}{N-i} \leq 1-\frac{d-1}{N-1} \ .$$

Therefore, we obtain that

$$\begin{aligned}&\Pr[X_d \geq 1] \\ &\geq 1-\left(1-\frac{d-1}{N-1}\right)^B \\ &= 1-\left(\left(1-\frac{d-1}{N-1}\right)^{N-1}\right)^{\frac{B}{N-1}} \\ &\geq 1-e^{-\frac{(d-1)B}{N-1}} \ .\end{aligned}$$

We bound the expression above by considering two cases:

- Case 1: $\frac{(d-1)B}{N-1} \leq 1.59$. For any $x \in [0, 1.59]$ it is known that $e^{-x} \leq 1 - \frac{x}{2}$. Applying this inequality we get that $\Pr[X_d \geq 1] \geq 1 - e^{-\frac{(d-1)B}{N-1}} \geq \frac{(d-1)B}{2(N-1)}$.
- Case 2: $\frac{(d-1)B}{N-1} > 1.59$. In this case, it holds that $\Pr[X_d \geq 1] \geq 1 - e^{-1.59} \geq 0.79$

Overall, in both cases we have $\Pr[X_d \geq 1] \geq \min\{0.5, \frac{(d-1)\cdot B}{2(N-1)}\}$. □

Given Claim 5.11, we are now ready to bound the success probability.

Proof of Claim 5.10. Assuming the first query to the matrix was 1-entry, A continues to sample entries from the same row. Note that for each row, the number of 1-entries may be different which affects the success probability of the algorithm. Therefore, we "divide" the rows into "buckets" by the number of 1-entries in it. Formally, for every $0 \leq d \leq N$, we define δ_d be the fraction of rows with exactly d 1-entries.

When $d \leq 1$, we know that the success probability is 0. Thus, we consider only δ_d for $d \geq 2$. This lets us derive the following:

$$\begin{aligned}
&\Pr[\mathsf{CollGame}(A, H) = 1] \\
&\geq \sum_{d=2}^{N} \delta_d \frac{d}{N} \cdot \Pr\left[X_d \geq 1\right] \\
&\geq \sum_{d=2}^{N} \delta_d \frac{d}{N} \cdot \left(\min\left\{\frac{1}{2}, \frac{(d-1)\cdot B}{2(N-1)}\right\}\right)
\end{aligned}$$

Let $n := \left\lfloor 1 + \frac{N-1}{B} \right\rfloor$, then,

$$\begin{aligned}
&\Pr[\mathsf{CollGame}(A, H) = 1] \\
&\geq \sum_{d=2}^{n} \delta_d \frac{d}{N} \cdot \frac{(d-1)\cdot B}{2(N-1)} + \sum_{d=n+1}^{N} \delta_d \frac{d}{N} \cdot \frac{1}{2} \\
&= \frac{B}{2} \sum_{d=2}^{n} \delta_d \frac{d(d-1)}{N(N-1)} + \frac{1}{2} \cdot \sum_{d=n+1}^{N} \delta_d \frac{d}{N} \\
&= \frac{B}{2N(N-1)} \sum_{d=0}^{n} \delta_d \cdot d(d-1) + \frac{1}{2} \cdot \sum_{d=n+1}^{N} \delta_d \frac{d}{N}
\end{aligned}$$

Let $\epsilon_1 := \sum_{d=0}^{n} \delta_d \frac{d}{N}$ and $\epsilon_2 := \sum_{d=n+1}^{N} \delta_d \frac{d}{N}$. By Jensen's inequality we get that

$$\frac{1}{N(N-1)} \sum_{d=0}^{n} \delta_d \cdot d(d-1) \geq \frac{1}{N(N-1)} \cdot \epsilon_1 N \left(\epsilon_1 N - 1\right) \geq \frac{\epsilon_1^2 \cdot N - \epsilon_1}{N} = \epsilon_1^2 - \frac{\epsilon_1}{N} \ .$$

Therefore we get,

$$\Pr[\mathsf{CollGame}(A, H) = 1] \geq \frac{B}{2}\left(\epsilon_1^2 - \frac{\epsilon_1}{N}\right) + \frac{1}{2}\epsilon_2 \ .$$

Since $\epsilon_1 + \epsilon_1 = \epsilon$, the minimum of the above expression is where $\epsilon_1 = \epsilon$. Thus, we can write

$$\Pr[\mathsf{CollGame}(A, H) = 1] \geq \frac{B}{2}\left(\epsilon^2 - \frac{\epsilon}{N}\right) \geq \frac{1}{2 \cdot 2\sqrt{\epsilon}} \cdot \epsilon^2 - \frac{\epsilon}{2 \cdot \sqrt{\epsilon} N} = \frac{\epsilon^{1.5}}{4} - \frac{\sqrt{\epsilon}}{2N} \ .$$

Since $\epsilon \geq \frac{4}{N}$, it holds that,

$$\frac{\sqrt{\epsilon}}{2N} \leq \frac{\sqrt{\epsilon}}{2\left(\frac{4}{\epsilon}\right)} = \frac{\epsilon^{1.5}}{8} \ .$$

This leads to,

$$\Pr[\mathsf{CollGame}(A,H)=1] \geq \frac{\epsilon^{1.5}}{8} \ ,$$

which completes the proof. □

□

5.3 Rogue soundness error bound from the collision game

We now use the algorithm for the collision game in order to construct an algorithm that extracts a witness $\mathbb{w}$ for an instance $\mathbb{x}$. Then, combined with the second-moment assumption we prove Theorem 5.6.

First, we prove the following lemma (which is interesting on its own):

Lemma 5.12. *Let $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda, \boldsymbol{K}), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K})$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let Π be an algebraic batch Σ-protocol for a relation $\mathcal{R}$ and let $\Pi^* = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be the batch Σ-protocol constructed from Π as defined in Construction 5.3. Let $t_{\boldsymbol{V}}$ denote the running time of the verifier $\boldsymbol{V}$ and let t_W denote the running time of the witness extractor. Let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ be a distribution over the relation where each $\mathcal{D}_\lambda$ produces $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$. For every prover $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2)$ that runs in time $t_{\tilde{\boldsymbol{P}}}$, there exists an algorithm A^* such that:*

1. $\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] \leq 2 \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)$.
2. $\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] \leq 4 \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)^2$.
3. *Either* $\epsilon < \frac{4}{|\mathcal{C}_{\mathsf{pp}}|}$ *or* $\Pr\left[(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \geq \frac{\epsilon^{1.5}}{8}$ *where ϵ is the rogue-soundness error of Π^* with respect to a distribution $\mathcal{D}$ and the setup algorithm* Setup.

Proof. We denote by aux the variable for tuples of $(\mathsf{pp}, \mathbb{x}, \vec{\beta})$ where $\vec{\beta} = (\beta_2, \ldots, \beta_k)$ and $\beta_i \in \{0,1\}^{\mathsf{r}}$. We consider binary matrices $H = \{H_{\mathsf{aux}}\}_{\mathsf{pp},\mathbb{x},\vec{\beta}} \in \{0,1\}^{R\times N}$, where the R rows correspond to $\tilde{\mathbf{P}}$'s randomness and the N columns correspond to $\mathbf{V}$'s randomness for one instance. Note that although $\tilde{\mathbf{P}}$'s and $\mathbf{V}$'s randomness depends on the number of instances that the prover outputs, we can always bound it by the randomness size when $\tilde{\mathbf{P}}$ outputs $\mathbf{K}$ instances.

An entry of H_{aux} equals 1 if and only if the corresponding transcript (between $\tilde{\mathbf{P}}$ and $\mathbf{V}$) is accepting. Recall that every algorithm A for the collision game aims to find $k+1$ entries with the value 1 in the same row. As $\tilde{\mathbf{P}}$'s randomness is fixed along one row, finding two 1-entries in the same row correspond to finding two accepting transcripts $(\alpha, \beta_1, \vec{\beta}, \gamma), (\alpha, \beta_1', \vec{\beta}, \gamma')$. Given Claim 5.5, Π^* has local special soundness, i.e., there exists an algorithm E that runs in time t_W which given two accepting transcripts as considered above, extracts a witness for the instance $\mathbb{x}_1$.

Let A be the algorithm for the collision game constructed in Lemma 5.7, we construct the algorithm A^* as follows:

$A^*(\mathsf{pp}, \mathbb{x}_1)$

1. Initialize an empty mapping M between the randomness used by $\tilde{\mathbf{P}}$ and $\mathbf{V}$ and the transcript between them.
2. Let r be $\mathbf{V}$'s randomness size for each instance. For $2 \leq i \leq \mathbf{K}$, sample $\beta_i \leftarrow \{0,1\}^{\mathsf{r}}$.
3. Invoke $A(1)$. When A performs a query on (ρ, β) answer as follows:
   (a) Invoke $((\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k), \alpha, \mathsf{st}) \leftarrow \tilde{\mathbf{P}}_1(\mathsf{pp}, \mathbb{x}_1; \rho)$.
   (b) Invoke $\gamma \leftarrow \tilde{\mathbf{P}}_2(\beta, \beta_2, \ldots, \beta_k, \mathsf{st})$.
   (c) Set $M[(\rho, \beta)] \leftarrow (\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \beta, \beta_2, \ldots, \beta_k, \gamma)$.
   (d) Return $\mathbf{V}(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \beta, \beta_2, \ldots, \beta_k, \gamma)$ as the answer to the query.
4. When A outputs ρ, β_1, β_2: set $(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_1^*, \beta_1^*, \beta_{1,2}^* \ldots, \beta_{1,k}^*, \gamma_1^*) \leftarrow M[\rho, \beta_1]$ and $(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_2^*, \beta_2^*, \beta_{2,2}^* \ldots, \beta_{2,k}^*, \gamma_2^*) \leftarrow M[\rho, \beta_2]$.
5. Run $\tilde{\mathbb{w}}_1 \leftarrow E(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_1^*, \beta_{1,2}^*, \ldots, \beta_{1,k}^*, (\beta_{1,1}^*, \gamma_{1,1}^*), (\beta_{2,1}^*, \gamma_{2,1}^*))$.
6. Output $\tilde{w}_1$.

We prove each claim separately.

Claim 5.13 (Expected running time)**.** *It holds that* $\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] \leq 2 \cdot (t_{\tilde{\boldsymbol{P}}} + \boldsymbol{V} + t_W)$.

Proof. Observe that whenever A query H, the algorithm A^* invokes $\tilde{\mathbf{P}}$ and $\mathbf{V}$. Thus, the expected number of invocations that A^* performs to $\tilde{\mathbf{P}}$ and $\mathbf{V}$ is the expected number of queries performed by A. Thus,

$$\begin{aligned}\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] &\leq \mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathbf{P}}} + \mathbf{V}) + t_W \\ &\leq 2 \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W) \ .\end{aligned}$$

□

Claim 5.14 (Second-moment of expected running time)**.** *It holds that* $\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] \leq 4 \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)^2$.

Proof. Following the same observation as in Claim 5.13 we obtain that

$$\begin{aligned}\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] &\leq \left(\mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}})\right)^2 + t_W^2 \\ &\leq \left(\mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)\right)^2 \\ &= \mathbb{E}\left[\mathcal{Q}_A\right]^2 \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)^2 \ .\end{aligned}$$

Jensen's inequality leads to

$$\begin{aligned}\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] &\leq \mathbb{E}\left[\mathcal{Q}_A^2\right] \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)^2 \\ &\leq 4(t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)^2 \ .\end{aligned}$$

□

Claim 5.15 (Success probability)**.** *Either* $\epsilon < \frac{4}{|\mathcal{C}_{\mathsf{pp}}|}$ *or* $\Pr\left[(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1)\end{array}\right] \geq \frac{\epsilon^{1.5}}{8}$ *where* ϵ *is the rogue-soundness error of* Π^* *with respect to a distribution* $\mathcal{D}$ *and the setup algorithm* Setup.

Proof. Whenever A succeeds in the collision game with H_{aux}, the algorithm A^* outputs a witness for $\mathbb{x}_1$. Thus,

$$\begin{aligned}&\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&=\sum_{\mathsf{aux}}\Pr[\mathsf{aux}]\cdot\Pr\left[\mathsf{CollGame}(A,H_{\mathsf{aux}})=1\right]\ .\end{aligned}$$

For every $\mathsf{aux}=(\mathsf{pp},\mathbb{x},\vec{\beta})$, we let

$$\epsilon_{\mathsf{aux}}=\Pr\left[\begin{array}{l|l}\mathbf{V}(\mathsf{pp},\mathbb{x},\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k,\alpha,\beta,\beta_2,\ldots,\beta_k,\gamma)=1 & ((\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k),\alpha,\mathsf{st})\leftarrow\tilde{\mathbf{P}}_1(1^\lambda,\mathsf{pp},\mathbb{x})\\\text{conditioned on }\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K}) & \beta_2,\ldots,\beta_k\leftarrow\mathcal{C}_{\mathsf{pp}}\\\wedge\,(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp}) & \gamma\leftarrow\tilde{\mathbf{P}}_2(\mathsf{st},\beta_2,\ldots,\beta_k)\\\wedge\,\beta_2,\ldots,\beta_k\leftarrow\mathcal{C}_{\mathsf{pp}} & \end{array}\right]\ .$$

The collision game matrix H_{aux} has ϵ_{aux} fraction of 1-entries. Thus, conditioned on aux, the probability that A succeeds in the collision game is $\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8}$. Therefore,

$$\begin{aligned}\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]&=\sum_{\mathsf{aux}}\Pr[\mathsf{aux}]\cdot\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8}\\&=\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8}\right]\\&\geq\frac{\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathsf{aux}}\right]^{1.5}}{8}&(4)\\&\geq\frac{\epsilon^{1.5}}{8}\ ,&(5)\end{aligned}$$

where Equation (4) follows from Jensen's inequality and Equation (5) follows from the fact that $\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathsf{aux}}\right]=\epsilon$. □

□

We are now ready to show a bound on the rogue soundness error of batch Σ-protocol defined in Construction 5.3.

Proof of Theorem 5.6. Let $\tilde{\mathbf{P}}$ be a cheating prover and let $\epsilon_{\mathcal{D}}$ be the rogue soundness error with respect to $\mathcal{D}$ and Setup. Given Lemma 5.12 and the assumption that $\mathcal{R}$ is second-moment hard with respect to the distribution $\mathcal{D}$ and the setup algorithm Setup, it holds that either $\epsilon_{\mathcal{D}}<\frac{4}{|\mathcal{C}_{\mathsf{pp}}|}$ or,

$$\begin{aligned}\frac{\epsilon_{\mathcal{D}}^{1.5}}{8}&\leq\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&\leq\frac{\Delta\cdot\mathbb{E}\left[T^2_{A^*,\mathcal{D}}\right]}{|\mathcal{W}_\lambda|^\omega}\\&\leq\frac{\Delta\cdot4\cdot(t_{\tilde{\mathbf{P}}}+t_{\mathbf{V}}+t_W)^2}{|\mathcal{W}_\lambda|^\omega}\ .\end{aligned}$$

This leads to

$$\epsilon_{\mathcal{D}} \leq \left(\frac{\Delta \cdot 32 \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} \right)^{2/3} .$$

Overall we derive the following bound

$$\begin{aligned} \epsilon_{\mathcal{D}} &\leq \max \left\{ \left(\frac{\Delta \cdot 32 \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} \right)^{2/3}, \frac{4}{|\mathcal{C}_{\mathsf{pp}}|} \right\} \\ &\leq \left(\frac{\Delta \cdot 32 \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} \right)^{2/3} + \frac{4}{|\mathcal{C}_{\mathsf{pp}}|} \end{aligned}$$

□

5.4 Algebraic batch identification schemes

An identification scheme consists of a Σ-protocol for relation $\mathcal{R}$ and an algorithm Gen that produces a distribution over $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ where the public key is the instance $\mathbb{x}$ and the secret key is the witness $\mathbb{w}$. Similarly, we construct a batch identification scheme that consists of batch Σ-protocol defined in Construction 5.3 and an algorithm Gen that given public parameters pp, produces a distribution over $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}(\mathsf{pp})$.

Note that the execution of ID is as the execution of the batch Σ-protocol where each public key pk corresponds to an instance, and a secret key sk corresponds to a witness.

We consider the rogue-security notion of batch identification scheme, asking a cheating prover $\tilde{\mathbf{P}}$ given as input an instance $\mathbb{x}$ produced by Gen, to convince the verifier $\mathbf{V}$ on $(\mathbb{x}, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k)$ where $\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k$ are adaptively chosen by $\tilde{\mathbf{P}}$ while given access to an honest transcript-generator for the instance $\mathbb{x}$ and another $(k-1)$ instances by its choice. Formally, we let $\mathsf{Trans}_{\mathsf{pk}_1, \mathsf{sk}_1}(\cdot)$ denote an oracle that when queried with input $(\mathsf{pk}_2, \mathsf{sk}_2), \ldots (\mathsf{pk}_k, \mathsf{sk}_k)$, runs an honest execution of the protocol on input $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots (\mathsf{pk}_k, \mathsf{sk}_k)$ and returns the resulting transcripts (α, β, γ). We define the rogue-security of a batch identification scheme as follows:

Definition 5.16 (Rogue soundness)**.** *Let* $\mathsf{ID} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a batch identification scheme for a relation* $\mathcal{R}$*. Then,* ID *is* (t, ϵ)*-rogue soundness (with respect to* Gen *and* Setup*) if for every* $\lambda, k \in \mathbb{N}$ *such that* $k \leq \boldsymbol{K}$ *and for any* t*-time malicious prover* $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2)$ *that performs* q *queries to the transcript-generation oracle it holds that:*

$$\Pr\left[\mathsf{StrongIdent}_{\mathsf{ID}}(\tilde{\boldsymbol{P}}, \lambda) \right] \leq \epsilon(\lambda, t, \mathsf{q}, \boldsymbol{K}) ,$$

where the experiment $\mathsf{StrongIdent}_{\mathsf{ID}}(\tilde{\boldsymbol{P}}, \lambda)$ *defined as follows:*

$\mathsf{StrongIdent}_{\mathsf{ID}}(\tilde{\mathbf{P}}, \lambda)$**:**

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K})$.
2. $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}(\mathsf{pp})$.
3. $((\tilde{\mathsf{pk}}_2, \ldots, \tilde{\mathsf{pk}}_k), \alpha, \mathsf{st}) \leftarrow \tilde{\boldsymbol{P}}_1^{\mathsf{Trans}_{\mathsf{pk}_1, \mathsf{sk}_1}(\cdot)}(\mathsf{pp}, \mathsf{pk}_1)$.
4. $\beta \leftarrow \mathcal{C}_{\mathsf{pp}}$.
5. $\gamma \leftarrow \tilde{\boldsymbol{P}}_2(\mathsf{st}, \beta)$.
6. *Output* $\boldsymbol{V}(\mathsf{pp}, \mathsf{pk}_1, \tilde{\mathsf{pk}}_2, \ldots, \tilde{\mathsf{pk}}_k, \alpha, \beta, \gamma) = 1$.

Recall that batch identification scheme ID consists of a batch Σ-protocol Π^* defined in Construction 5.3 such that the execution of ID is as the execution of Π^* where each public key pk corresponds to an instance and a secret key sk corresponds to a witness. Thus, if Π^* is zero-knowledge, we can assume that every malicious prover does not query the transcript-generation oracle, as such queries can be internally simulated given

the public keys. Formally, if Π^* is t-time zero-knowledge (Definition 3.8), for every malicious prover that performs q queries to the transcript-generation oracle $\mathsf{Trans}_{\mathsf{pk}_1,\mathsf{sk}_1}(\cdot)$, we can construct a malicious prover that does not query the transcript-generation oracle and instead runs the simulator q times to generate transcripts. Specifically, if Π^* has t_{Sim}-time zero-knowledge, any malicious prover that runs in time $t_{\tilde{\boldsymbol{P}}}$ and performs q queries to $\mathsf{Trans}_{\mathsf{pk}_1,\mathsf{sk}_1}(\cdot)$, can be simulated by a malicious prover that runs in time $t_{\tilde{\boldsymbol{P}}} + \mathsf{q} \cdot t_{\mathsf{Sim}}$.

Recall that every batch Σ-protocol Π^* defined in Construction 5.3 is constructed from an algebraic Σ-protocol Π. We now show that if Π is t_{Sim}-time zero-knowledge, then Π^* is $(k \cdot t_{\mathsf{Sim}})$-zero-knowledge. Formally, we prove the following.

Claim 5.17. *Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be an algebraic Σ-protocol for a relation $\mathcal{R}$ and let Π^* be the batch Σ-protocol constructed from Π as defined in Construction 5.3 with a bound $\boldsymbol{K}$ on the number of instances. If Π is t_{Sim}-time zero-knowledge, then Π^* is $(\boldsymbol{K} \cdot t_{\mathsf{Sim}})$-time zero-knowledge.*

Proof. Let Sim be a simulator for the distributions of Π. Consider the algorithm Sim^* which takes as input public parameters pp and instances $\mathbb{x}_1, \ldots, \mathbb{x}_k$ and is defined as follows:

1. For every $i \in [k]$: invoke $(\alpha_i, \beta_i, \gamma_i) \leftarrow \mathsf{Sim}(\mathsf{pp}, \mathbb{x}_i)$.
2. Compute $\alpha = \Pi_{i=1}^{k} \alpha_i$ and $\gamma = \sum_{i=1}^{k} \gamma_i$.
3. Output $(\alpha, \beta_1, \ldots, \beta_k, \gamma)$.

Observe that the distribution of Sim^*'s outputs is identical to the distribution of the transcript of an honest execution of Π^*. This claim can be proven using a simple hybrid argument. □

Thus, from Theorem 5.6 we derive the following corollary:

Corollary 5.18. *Let $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K}), t_{\mathsf{Sim}} = t_{\mathsf{Sim}}(\lambda, \boldsymbol{K}), \mathsf{q} = \mathsf{q}(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let Π be an algebraic Σ-protocol for relation $\mathcal{R}$ with t_{Sim}-time zero-knowledge and let $\Pi^* = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be the batch Σ-protocol constructed from Π as defined in Construction 5.3. Let $\mathsf{ID} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be the batch identification scheme consists with Π^*. If $\mathcal{R}$ is second-moment hard with respect to Gen, then for any malicious prover $\tilde{\boldsymbol{P}}$ that runs in time $t_{\tilde{\boldsymbol{P}}}$ and issues q transcript-generation queries it holds that*

$$\Pr\left[\mathsf{StrongIdent}_{\mathsf{ID}}(\tilde{\boldsymbol{P}}, \lambda)\right] \leq \left(\frac{\Delta \cdot 32 \cdot (t_{\tilde{\boldsymbol{P}}} + \mathsf{q} \cdot \boldsymbol{K} \cdot t_{\mathsf{Sim}} + t_{\boldsymbol{V}} + t_W)^2}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3} + \frac{4}{|\mathcal{C}_{\mathsf{pp}}|} \ ,$$

where $t_{\boldsymbol{V}}$ denotes the running time of the verifier $\boldsymbol{V}$ and t_W denotes the running time of the witness extractor.

6 Non-interactive batch arguments from algebraic Sigma protocols

In the previous section, we constructed batch Σ-protocols from algebraic Σ-protocols and bounded their rogue-soundness error. In this section, we construct non-interactive batch arguments from algebraic Σ-protocols via the Fiat-Shamir paradigm [FS86]. Similarly, the goal of this section is to bound the rogue-instance security of the non-interactive analog.

In Section 6.1 we construct non-interactive batch arguments from algebraic Σ-protocols. As in the previous section, in Section 6.2 we introduce a tree game, then in Section 6.3, we show the rogue-instance security of non-interactive batch arguments constructed via the Fiat-Shamir paradigm from algebraic Σ-protocols.

6.1 Non-interactive batch arguments

In this section, we construct the non-interactive analog of batch Σ-protocols defined in Construction 5.3.

Recall that in the construction of batch Σ-protocols, the prover is given as input k different challenges for each input. We wish to keep this property in the non-interactive analog, formally, we consider the following:

Construction 6.1 (NARGs from batch Σ-protocols)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a relation, where $\mathcal{R}_\lambda \subseteq \mathcal{P}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ for any $\lambda \in \mathbb{N}$ and let $\boldsymbol{K} \in \mathbb{N}$ be a bound on the number of instances. Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be an algebraic Σ-protocol with an algebraic one-way function $(\mathsf{Setup}, \mathsf{F})$. We define $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ to be the non-interactive batch argument constructed as follows for every $k \leq \boldsymbol{K}$. The argument prover $\mathcal{P}$ receives as input public parameters pp, instances $\mathbb{x}_1, \ldots, \mathbb{x}_k$ and witnesses $\mathbb{w}_1, \ldots, \mathbb{w}_k$, and the verifier $\mathcal{V}$ receives as input the public parameters pp, the instances $\mathbb{x}_1, \ldots, \mathbb{x}_k$ and an argument string π. Both receive query access to a random oracle $f \in \mathcal{U}(\lambda)$ that outputs λ bits.*

- $\mathcal{P}^f(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k))$*:*
  1. *For every $i \in [k]$, invoke $(R_i, \mathsf{st}_i) \leftarrow \boldsymbol{P}_1(\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i)$.*
  2. *Compute $\alpha = \Pi_{i=1}^k R_i$.*
  3. *For every $i \in [k]$: derive the challenge $\beta_i := f(\mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, i)$.*
  4. *For every $i \in [k]$: invoke $\gamma_i \leftarrow \boldsymbol{P}_2(\mathsf{st}_i, \beta_i)$.*
  5. *Compute $\gamma = \sum_{i=1}^k \gamma_i$.*
  6. *Output the argument string $\pi := (\alpha, \gamma)$.*
- $\mathcal{V}^f(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \pi)$*:*
  1. *Parse the argument Π as a tuple (α, γ).*
  2. *For every $i \in [k]$: derive the challenge $\beta_i := f(\mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, i)$.*
  3. *Check that $\mathsf{F}(\gamma) \stackrel{?}{=} \alpha \cdot \Pi_{i=1}^k \mathbb{x}_i^{\beta_i}$.*

The setup algorithm Setup *remains the same.*

Recall that we showed in Claim 5.5 that the batch Σ-protocol defined in Construction 5.3 has local special soundness. Similarly, it is easy to see that the claim holds for the non-interactive batch argument constructed in Construction 6.1.

In Section 6.3 we present a concrete bound on the rogue-soundness error of the non-interactive batch argument NARG defined in Construction 6.1. Specifically, we formally establish the following theorem:

Theorem 6.2. *Let $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\mathcal{P}}} = \tilde{\mathcal{P}}(\lambda, \boldsymbol{K}), t_{\mathcal{V}} = t_{\mathcal{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K}), \mathsf{q} = \mathsf{q}(\lambda, \boldsymbol{K})$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be an algebraic Σ-protocol for a relation $\mathcal{R}$ and let $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be the non-interactive batch argument constructed from Π as defined in Construction 6.1. If $\mathcal{R}$ is second-moment hard*

with respect to a distribution $\mathcal{D}$ and the setup algorithm Setup*, then* NARG *has $(t_{\tilde{\mathcal{P}}}, \mathsf{q}, \epsilon)$-rogue soundness error such that*

$$\epsilon_{\mathcal{D}}(\lambda, t_{\tilde{\mathcal{P}}}, t_{\mathcal{V}}, t_W, \mathsf{q}, \boldsymbol{K}) \leq \left(\frac{\Delta \cdot 32\boldsymbol{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)^2}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3} + \frac{4\boldsymbol{K} \cdot \mathsf{q}}{2^\lambda} \ ,$$

where $t_{\mathcal{V}}$ denotes the running time of the verifier $\mathcal{V}$ and t_W denotes the running time of the witness extractor.

6.2 The tree game

We introduce a tree game where an algorithm is given oracle access to a tree T where the value of each leaf is a number. Informally, an algorithm succeeds in the game if it finds $k+1$ leaves with the same value in the same sub-tree.

Consider a complete tree $T = (V, E)$ of depth l and degree r. Let $\mathsf{Leaves}(T)$ be the leaves of T and let $\mathsf{val}\colon \mathsf{Leaves}(T) \to \{0, \ldots, l-1\}$ be a function represents the value of each leaf in T. Note that not all leaves hold a number value, we consider the value of such a leaf as $\perp$. Let $\mathsf{lca}()$ be a function that describes the lowest common ancestor of a group of nodes.

During the execution of the game, the algorithm A is given as input a number k and oracle access to the tree T. Its goal is to find $k+1$ leaves $u_1, \ldots, u_{k+1}$ with the same value i that have the same lowest common ancestor v such that $\mathsf{height}(v) = i$. More precisely, algorithm A knows the tree's structure and is given oracle access to the value of each node in T. We say that A succeeds in the game if it produces $k+1$ leaves with the same value i that have the same lowest common ancestor v such that $\mathsf{height}(v) = i$. Formally, the game is constructed as follows:

$\mathsf{TreeCollGame}_k(A, T)$

1. The algorithm $A(k)$ is given oracle access to T and outputs $u_1, \ldots, u_{k+1}$.
2. The output of the game is 1 if and only if $u_1, \ldots, u_{k+1} \in \mathsf{Leaves}(T)$ and there exists an internal node $v \in T \setminus \mathsf{Leaves}(T)$ such that $\mathsf{lca}(u_1, \ldots, u_{k+1}) = v$ and $\mathsf{height}(v) = \mathsf{val}(u_1) = \ldots = \mathsf{val}(u_{k+1})$.

In particular, in this section, we refer to the tree game when $k = 1$. We construct an algorithm that finds two leaves with the same value in T with probability at least $\approx \epsilon^{3/2}/l$ and performs ≈ 2 queries to T where ϵ is the fraction of non-bot leaves in T and l is the depth of T. Formally, we prove the following.

Lemma 6.3. *Let $T = (V, E)$ be a complete tree of depth l and degree r and let $\mathsf{val}\colon \mathsf{Leaves}(T) \to \{0, \ldots, l-1\}$ be a function represents the value of each leaf in T. Let ϵ be the fraction of non-bot leaves in T and let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to T. Then, there exists an algorithm A with oracle access to T such that on input $k = 1$ the following holds:*

1. $\mathbb{E}[\mathcal{Q}_A] \leq 2$.
2. $\mathbb{E}\left[\mathcal{Q}_A^2\right] \leq 4$.
3. *Either $\epsilon < \frac{4l}{r}$ or $\Pr[\mathsf{TreeCollGame}(A, T) = 1] \geq \frac{\epsilon^{1.5}}{8l}$.*

Proof. Let $p\colon \{0, \ldots, l\} \times \mathsf{Leaves}(T) \to V$ be a function such that $\mathsf{parent}(i, u) = v$ if v is the parent of u at level i in T. Let T_v to denote the sub-tree rooted in v and let $B = \left\lceil \frac{1}{\sqrt{\epsilon}} - 1 \right\rceil$. We consider the following algorithm A:

$A^T(1)$

1. Sample $u \leftarrow \mathsf{Leaves}(T)$. If $\mathsf{val}(u) = \perp$ abort. Let $v = \mathsf{parent}(\mathsf{val}(u), u)$ and $T' = T_{\mathsf{parent}(\mathsf{val}(u)-1,u)}$.
2. Let $S = \emptyset$. For every $i \in [B]$: sample $u_i \leftarrow \mathsf{Leaves}(T_v \setminus \{S \cup T'\})$ and set $S = S \cup \{u_i\}$.
3. If for every $i \in [B]$ it holds that $\mathsf{val}(u_i) \neq \mathsf{val}(u)$, abort.
4. Choose uniformly at random an index i such that $\mathsf{val}(u_i) = \mathsf{val}(u)$. Return u, u_i.

We now prove each claim separately.

Claim 6.4. *It holds that* $\mathbb{E}[\mathcal{Q}_A] \leq 2$.

Proof. By the description of A, it performs at least one query. Then, with probability ϵ it continues to sample B leaves, therefore,

$$\mathbb{E}[\mathcal{Q}_A] = 1 + \epsilon \cdot B \leq 1 + \sqrt{\epsilon} \leq 2 \ .$$

□

Claim 6.5. *It holds that* $\mathbb{E}\left[\mathcal{Q}_A^2\right] \leq 4$.

Proof. By the description of A, with probability $1 - \epsilon$, it performs a single query, and with probability ϵ it performs $(1 + B)$ queries. Thus, we can bound the expectation squared by

$$\begin{aligned}
\mathbb{E}\left[\mathcal{Q}_A^2\right] &= (1-\epsilon) \cdot 1^2 + \epsilon \cdot (1+B)^2 \\
&= 1 - \epsilon + \epsilon(1 + 2B + B^2) \\
&= 1 + 2\epsilon B + \epsilon B^2 \\
&\leq 1 + 2\sqrt{\epsilon} + 1 \\
&\leq 4 \ .
\end{aligned}$$

□

Claim 6.6. *Either* $\epsilon < \frac{4l}{r}$ *or* $\Pr[\mathsf{TreeCollGame}(A, T) = 1] \geq \frac{\epsilon^{1.5}}{8l}$.

Proof. We begin with a short high-level overview of the proof. Assume A samples a leaf u with the value h such that $\mathsf{parent}(h, u) = v$ for some leaf v and a value h. Then, A continues to sample leaves from T_v in order to find another leaf with the value h. Note that for every h and v, the number of leaves with the value h in T_v may be different, which affects its success probability. Therefore, for every value h, we "divide" the internal nodes to "buckets" by the probability to sample a leaf with the value h in its sub-tree, and then we look at the probability to "reach" each bucket.

Formally, for every $0 \leq d \leq l \log r$ and $0 \leq h \leq l - 1$, we let

$$\delta_{d,h} = \Pr_{v:\mathsf{height}(v)=h}\left[\frac{|\{u \in \mathsf{Leaves}(T_v) : \mathsf{val}(u) = h\}|}{|\mathsf{Leaves}(T_v)|} \in \left[2^{-d}, 2^{-d+1}\right]\right] \ .$$

Note that a node v is in the d-th "bucket" if the probability to sample a leaf with the value h in the sub-tree T_v is in $\left[2^{-d}, 2^{-d+1}\right]$.

For every $0 \leq i \leq l - 1$, let A_i be the same algorithm as A except for one minor change, instead of aborting in Item 1 if $\mathsf{val}(u) = \perp$, it aborts if $\mathsf{val}(u) \neq i$. By the description of A it holds that,

$$\Pr[\mathsf{TreeCollGame}(A, T) = 1] = \sum_{i=0}^{l-1} \Pr[\mathsf{TreeCollGame}(A_i, T) = 1] \ .$$

We now bound the above term $\Pr[\mathsf{TreeCollGame}(A_i, T) = 1]$. Let u be the first leaf A samples and let $v = \mathsf{parent}(\mathsf{val}(u), u)$. We say that a leaf $u_i \in T_v$ is "good" if $\mathsf{val}(u_i) = \mathsf{height}(v)$. By the definition of $\delta_{d,h}$, there exists a d such that the fraction of good leaves in T_v is at least 2^{-d}. Thus, the number of good leaves in T_v is at least $2^{-d} \cdot r^{l-h}$. Assuming $\mathsf{val}(u) = h$, algorithm A continues to sample leaves from $T_v \setminus T'$. Let η be the fraction of good leaves in $T_v \setminus T'$. Note that the number of good leaves in T' is at most r^{l-h-1}, therefore, the fraction of good leaves in $T_v \setminus T'$ is at least,

$$\begin{aligned}
\eta &\geq \frac{2^{-d} \cdot r^{l-h} - r^{l-h-1}}{r^{l-h} - r^{l-h-1}} \\
&= \frac{r^{l-h-1} \cdot (2^{-d} r - 1)}{r^{l-h-1} \cdot (r-1)} \\
&= \frac{2^{-d} r - 1}{r - 1} \\
&\geq 2^{-d} - \frac{1}{r} \ .
\end{aligned}$$

Let X be the number of good leaves A finds with B draws. By Claim 5.11, it holds that $\Pr[X \geq 1] \geq \min\left\{\frac{1}{2}, \eta \cdot \frac{B}{2}\right\}$. Therefore,

$$\begin{aligned}
&\Pr[\mathsf{TreeCollGame}(A_i, T) = 1] \\
&\geq \sum_{d=0}^{l \log r} \delta_{d,i} \cdot 2^{-d} \cdot \Pr[X \geq 1] \\
&\geq \sum_{d=0}^{l \log r} \delta_{d,i} \cdot 2^{-d} \cdot \min\left\{\frac{1}{2}, \frac{B}{2}\left(2^d - \frac{1}{r}\right)\right\}
\end{aligned}$$

Let $n := \left\lfloor \log\left(\frac{rB}{r+B}\right) \right\rfloor$, then,

$$\begin{aligned}
&\Pr[\mathsf{TreeCollGame}(A_i, T) = 1] \\
&\geq \sum_{d=0}^{n} \delta_{d,i} \cdot 2^{-d} \cdot \frac{1}{2} + \sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot 2^{-d} \cdot \frac{B}{2}\left(2^d - \frac{1}{r}\right) \\
&= \frac{1}{2} \cdot \sum_{d=0}^{n} \delta_{d,i} \cdot 2^{-d} + \frac{B}{2} \cdot \left(\sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot 2^{-2d} - \frac{1}{r} \cdot \sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot 2^{-d} \right) \ .
\end{aligned}$$

Let ϵ_i be the fraction of leaves with the value i. Let $\epsilon_{i,1} := \sum_{d=0}^{n} \delta_{d,i} \cdot 2^{-d}$ and $\epsilon_{i,2} := \sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot 2^{-d}$. By Jensen's inequality it holds that

$$\begin{aligned}
&\sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot 2^{-2d} \\
&= \sum_{d=n+1}^{l \log r} \delta_{d,i} \cdot \left(2^{-d}\right)^2 \\
&\geq \epsilon_{i,2}^2 \ .
\end{aligned}$$

Therefore,

$$\Pr[\mathsf{TreeCollGame}(A_i, T) = 1] \geq \frac{\epsilon_{i,1}}{2} + \frac{B}{2} \cdot \epsilon_{i,2}^2 - \frac{B}{2r} \cdot \epsilon_{i,2} \ .$$

Note that $\sum_{d=0}^{l \log r} \delta_{d,i} \cdot 2^{-d} = \epsilon_{i,1} + \epsilon_{i,2} \leq \epsilon$. Therefore, the minimum of the above expression is where $\epsilon_{i,2} = \epsilon_i$, which lets us write

$$\Pr[\mathsf{TreeCollGame}(A_i, T) = 1] \geq \frac{B}{2} \cdot \epsilon_i^2 - \frac{B}{2r} \cdot \epsilon_i \ .$$

Overall, we obtain,

$$\Pr[\mathsf{TreeCollGame}(A, T) = 1] \geq \sum_{i=0}^{l-1} \left(\frac{B}{2} \cdot \epsilon_i^2 - \frac{B}{2r} \cdot \epsilon_i \right) = \frac{B}{2} \cdot \sum_{i=0}^{l-1} \epsilon_i^2 - \frac{B}{2r} \sum_{i=0}^{l-1} \epsilon_i \ .$$

The following claim (which we prove later on) provides a lower bound on the above term $\sum_{n=0}^{l-1} \epsilon_n^2$.

Claim 6.7. *For every $p_1, \ldots, p_n$ such that $\sum_{i=1}^{n} p_i = \epsilon$ it holds that $\sum_{i=0}^{n} p_i^2 \geq \frac{\epsilon^2}{n}$.*

Given Claim 6.7 and the fact that $\sum_{i=0}^{l-1} \epsilon_i = \epsilon$, it holds that,

$$\begin{aligned}
&\Pr[\mathsf{TreeCollGame}(A, T) = 1] \\
&\geq \frac{B \cdot \epsilon^2}{2l} - \frac{B\epsilon}{2r} \\
&\geq \frac{1}{2\sqrt{\epsilon}} \cdot \frac{\epsilon^2}{2l} - \frac{1}{\sqrt{\epsilon}} \cdot \frac{\epsilon}{2r} \\
&= \frac{\epsilon^{1.5}}{4l} - \frac{\sqrt{\epsilon}}{2r} \ .
\end{aligned}$$

Since, $\epsilon \geq \frac{4l}{r}$, it holds that

$$\frac{\sqrt{\epsilon}}{2r} \leq \frac{\sqrt{\epsilon}}{2 \cdot \frac{4l}{\epsilon}} = \frac{\epsilon^{1.5}}{8l} \ .$$

This leads to,

$$\Pr[\mathsf{TreeCollGame}(A, T) = 1] \geq \frac{\epsilon^{1.5}}{8l} \ .$$

For completeness, we now prove Claim 6.7.

Proof of Claim 6.7. Observe that $\left(p_i - \frac{\epsilon}{n}\right)^2 = p_i^2 - \frac{2\epsilon \cdot p_i}{n} + \frac{\epsilon^2}{n^2}$, therefore,

$$\begin{aligned}
&\sum_{i=0}^{n} \left(p_i - \frac{\epsilon}{n}\right)^2 \\
&= \sum_{i=0}^{n} \left(p_i^2 - \frac{2\epsilon \cdot p_i}{n} + \frac{\epsilon^2}{n^2} \right) \\
&= \sum_{i=0}^{n} p_i^2 - \frac{2\epsilon}{n} \cdot \sum_{i=0}^{n} p_i + n \cdot \frac{\epsilon^2}{n^2} \\
&= \sum_{i=0}^{n} p_i^2 - \frac{2\epsilon}{n} \cdot \epsilon + \frac{\epsilon^2}{n} \ , \qquad (6)
\end{aligned}$$

where Equation (6) follows from the fact that $\sum_{i=0}^{n} p_i = \epsilon$. The above equation holds if and only if the following holds,

$$\sum_{i=0}^{n} p_i^2 = \sum_{i=0}^{n} \left(p_i - \frac{\epsilon}{n}\right)^2 + \frac{2\epsilon^2}{n} - \frac{\epsilon^2}{n} \geq \frac{\epsilon^2}{n} \ ,$$

which completes the proof. □

□

□

6.3 Rogue security for NARGs from algebraic batch Sigma protocols

We now show how to construct an algorithm that given an instance $\mathbb{x}$ extracts a corresponding witness $\mathbb{w}$ given the algorithm for the tree game. Then, combined with the second-moment assumption we prove Theorem 6.2.

First, we prove the following lemma (which is interesting on its own),

Lemma 6.8. *Let $t_{\tilde{\mathcal{P}}} = \tilde{\mathcal{P}}(\lambda, \boldsymbol{K}), t_{\mathcal{V}} = t_{\mathcal{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K}), \mathsf{q} = \mathsf{q}(\lambda, \boldsymbol{K})$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let $\Pi = (\mathsf{Setup}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be an algebraic Σ-protocol for a relation $\mathcal{R}$ and let $\mathsf{NARG} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be the non-interactive batch argument constructed from Π as defined in Construction 6.1. Let $t_{\mathcal{V}}$ denote the running time of the verifier $\boldsymbol{V}$ and let t_W denote the running time of the witness extractor. Let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ be a distribution over the relation where each $\mathcal{D}_\lambda$ produces $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$. For every cheating prover $\tilde{\mathcal{P}}$ that runs in time $t_{\tilde{\mathcal{P}}}$ and issues q queries to a random oracle f that outputs λ bits, there exists an algorithm A^* such that:*

1. $\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] \leq 2(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)$.
2. $\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] \leq 4(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)^2$.
3. *Either* $\epsilon \leq \frac{4\boldsymbol{K}\cdot\mathsf{q}}{2^\lambda}$ *or* $\Pr\left[(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \geq \frac{\epsilon^{1.5}}{8\boldsymbol{K}\cdot\mathsf{q}}$ *where ϵ is the rogue-soundness error of* NARG *with respect to a distribution $\mathcal{D}$ and the setup algorithm* Setup.

Proof. For simplicity, let $\tilde{\mathcal{P}}'$ be the same prover as $\tilde{\mathcal{P}}$ except for one minor change. Each time $\tilde{\mathcal{P}}$ queries f with $(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, 1)$, $\tilde{\mathcal{P}}'$ queries $f(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, i)$ for every $2 \leq i \leq k$, and only then queries $f(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, 1)$. Therefore, if $\tilde{\mathcal{P}}$ performs q queries, then $\tilde{\mathcal{P}}'$ performs at most $\mathsf{q}' = \mathsf{q} \cdot \mathbf{K}$.

We denote by aux the variable for tuples of $(\mathsf{pp}, \mathbb{x}, \rho)$ where $\rho \in \{0,1\}^{\mathsf{r}}$. We consider complete trees $T = \{T_{\mathsf{aux}}\}_{\mathsf{pp},\mathbb{x},\rho}$ of depth q' and degree 2^λ. The depth corresponds to the number of queries $\tilde{\mathcal{P}}'$ performs and the degree corresponds to the number of possible answers of f. We represent a leaf by all the answers from the random oracle that led to that output.

Note that an execution of $\tilde{\mathcal{P}}'$ corresponds to a random walk on the tree T_{aux}, therefore, a leaf $(y_1, \ldots y_{\mathsf{q}}) \in \mathsf{Leaves}(T)$ leads to an output $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_i, \gamma_i)$. We set the value of such a leaf to be the index of the random oracle query (starting from 0) in which $\tilde{\mathcal{P}}'$ queried f with $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_i, 1)$. If $\tilde{\mathcal{P}}'$ does not query $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_i, 1)$ or $\mathcal{V}(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_i, f(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_i, 1), \ldots, f(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, k), \gamma_i) = 0$, we set the value to be $\perp$.

Recall that every algorithm A for the tree game on input k, aims to find $k+1$ leaves with the same index i that have the same lowest common ancestor v such that $\mathsf{height}(v) = i$. As $\tilde{\mathcal{P}}'$'s randomness is fixed, finding $k+1$ non-bot leaves that point to the same node corresponds to finding $k+1$ accepting transcripts with a common first message and instances and distinct pairwise challenges for the first instance. More precisely, let A be the algorithm constructed in Lemma 6.3, algorithm A^* invokes A on input $k = 1$ while simulating its oracle queries to T by executing $\tilde{\mathcal{P}}'$ and $\mathbf{V}$. If A outputs two leaves, algorithm A^* creates two transcripts with a common first message and pairwise distinct challenges for the first instance. Recall that we showed in Claim 5.5 that the non-interactive batch argument defined in Construction 6.1 has local special soundness. Let $\mathsf{NARG.E}$ be the witness extractor of NARG.

Formally, we construct the algorithm A^* as follows:

$A^*(\mathsf{pp}, \mathbb{x}_1)$

1. Let r be $\tilde{\mathcal{P}}'$'s randomness size and sample $\rho \leftarrow \{0,1\}^{\mathsf{r}}$. Let M be a mapping between the answers from the random oracle to $\tilde{\mathcal{P}}'$'s output.
2. Invoke $A(1)$. When A performs a query on a leaf u do as follows:
   (a) Let $y_1, \ldots, y_{\mathsf{q}'}$ be the representation of u.
   (b) Invoke $(\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \gamma) \leftarrow \tilde{\mathcal{P}}'^f(\mathsf{pp}, \mathbb{x}_1; \rho)$. When $\tilde{\mathcal{P}}'$ query it's j-th query to f respond with y_j.
   (c) For every $i \in [k]$: let t_i denote the index of the query in which $\tilde{\mathcal{P}}'$ queried $f(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha_1, i)$. If $\tilde{\mathcal{P}}'$ does not query f with $(\mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, i)$, let $t_i = 0$.
   (d) If $t_1 = 0$ or $\mathbf{V}(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, y_{t_1}, \ldots, y_{t_k}, \gamma) = 0$, return $\perp$ as the answer to the query. Otherwise, return $(t_1 - 1)$ as the answer to the query.
   (e) Set $M[(y_1, \ldots, y_{\mathsf{q}'})] \leftarrow (\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, y_{t_1}, \ldots, y_{t_k}, \gamma)$.
3. When A outputs u_1, u_2:
   (a) Let $y_{i,1}, \ldots, y_{i,\mathsf{q}'}$ be the representation of u_i.
   (b) Set $(\tilde{\mathbb{x}}_{i,2}, \ldots, \tilde{\mathbb{x}}_{i,k_i}, \alpha_i^*, \beta_{i,1}^*, \ldots, \beta_{i,k_i}^* \gamma_i^*) = M[y_{i,1}, \ldots, y_{i,\mathsf{q}'}]$.
4. If $k_1 \neq k_2$, or $\alpha_1^* \neq \alpha_2^*$, or $(\tilde{\mathbb{x}}_{1,2}, \ldots, \tilde{\mathbb{x}}_{1,k_1}) \neq (\tilde{\mathbb{x}}_{2,2}, \ldots, \tilde{\mathbb{x}}_{2,k_2})$, or $\beta_{1,1}^* = \beta_{2,1}^*$, or there exists an index j such that $\beta_{1,j}^* \neq \beta_{2,j}^*$, abort.
5. Run $\tilde{\mathbb{w}}_1 \leftarrow \mathsf{NARG.E}(\alpha_1, \beta_{1,2}^*, \ldots, \beta_{1,k}^*, (\beta_1^*, \gamma_1^*), (\beta_2^*, \gamma_2^*))$.
6. Output $\tilde{\mathbb{w}}_1$.

We now prove each of the claims separately.

Claim 6.9 (Bound on the expected running time)**.** *It holds that* $\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] \leq 2(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)$.

Proof. Observe that whenever A query T, the algorithm A^* invokes $\tilde{\mathcal{P}}'$ and $\mathbf{V}$. Thus, the expected number of invocations that A^* performs to $\tilde{\mathcal{P}}'$ and $\mathbf{V}$ is the expected number of queries performed by A. Note that $\tilde{\mathcal{P}}'$'s running time is $t_{\tilde{\mathcal{P}}}$ and $\mathbf{V}$'s running time is at most $t_{\mathcal{V}}$, then,

$$\begin{aligned}\mathbb{E}\left[T_{A^*,\mathcal{D}_\lambda}\right] &\leq \mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}}) + t_W \\ &\leq 2(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W) \ . \end{aligned} \tag{7}$$

where Equation (7) follows from Lemma 6.3. □

Claim 6.10 (Bound on the second-moment of the expected running time)**.** *It holds that* $\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] \leq 4(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)^2$.

Proof. Following the same observation as in Claim 6.9 we obtain that

$$\begin{aligned}\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] &\leq \left(\mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}})\right)^2 + t_W^2 \\ &\leq \left(\mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)\right)^2 \\ &= \mathbb{E}\left[\mathcal{Q}_A\right]^2 \cdot (t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)^2 \ . \end{aligned}$$

Jensen's inequality leads to

$$\begin{aligned}\mathbb{E}\left[T^2_{A^*,\mathcal{D}_\lambda}\right] &\leq \mathbb{E}\left[\mathcal{Q}_A^2\right]\cdot(t_{\tilde{\mathcal{P}}}+t_{\mathcal{V}}+t_W)^2\\ &\leq 4(t_{\tilde{\mathcal{P}}}+t_{\mathcal{V}}+t_W)^2 \ .\end{aligned}$$

□

We now bound the success probability.

Claim 6.11. *Either* $\epsilon \leq \frac{4\boldsymbol{K}\cdot\mathsf{q}}{2^\lambda}$ *or* $\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\boldsymbol{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\geq\frac{\epsilon^{1.5}}{8\boldsymbol{K}\cdot\mathsf{q}}$ *where* ϵ *is the rogue-soundness error of* NARG *with respect to a distribution* $\mathcal{D}$ *and the setup algorithm* Setup.

Proof. Recall that A^* simulates A's oracle queries to T such that each leaf leads to an output of $\tilde{\mathcal{P}}'$. The value of each leaf that leads to an output $(\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_{k_i},\alpha_i,\gamma_i)$ is the index of the query in which $\tilde{\mathcal{P}}'$ queried f with $(\mathbb{x}_1,\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_{k_i},\alpha_i,1)$. If $\tilde{\mathcal{P}}'$ queried f with $(\mathbb{x}_1,\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_{k_i},\alpha_i,1)$, then it already queried $(\mathbb{x}_1,\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_{k_i},\alpha_i,j)$ for every $2\leq j\leq k$.

Whenever A succeeds in the tree game with T_{aux}, it outputs u_1,u_2 with the same value i that have the same lowest common ancestor v such that $\mathsf{height}(v)=i$. Since each leaf corresponds to $\tilde{\mathcal{P}}'$'s output, whenever A succeeds in the tree game it produces two outputs that queried to f by $\tilde{\mathcal{P}}'$ on the same query and answered differently. In fact, A produces two outputs which share the first message $(\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k,\alpha)$ and the $k-1$ last randomness, but differ on their first randomness. Thus,

$$\begin{aligned}&\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\\wedge\,(\mathbb{x},\mathbb{w})\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&=\sum_{\mathsf{aux}}\Pr[\mathsf{aux}]\cdot\Pr\left[\mathsf{TreeCollGame}(A,T_{\mathsf{aux}})=1\right]\end{aligned}$$

For every $\mathsf{aux}=(\mathsf{pp},\mathbb{x},\rho)$, we let

$$\epsilon_{\mathsf{aux}}=\Pr\left[\begin{array}{l}\mathcal{V}^f(\mathsf{pp},\mathbb{x},\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k,\alpha,\gamma)=1\\\text{conditioned on }\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\\wedge\,(\mathbb{x},\mathbb{w})\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\wedge\,\rho\leftarrow\{0,1\}^{\mathsf{r}}\end{array}\;\middle|\;(\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k,\alpha,\gamma)\leftarrow\tilde{\mathcal{P}}'^f(\mathsf{pp},\mathbb{x};\rho)\right] \ .$$

The tree T_{aux} has ϵ_{aux} fraction of non-bot leaves. Thus, conditioned on aux, the probability that A succeeds in the tree game is $\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8\cdot\mathsf{q}'}$. Therefore,

$$\begin{aligned}&\Pr\left[(\mathsf{pp},\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\\wedge\,(\mathbb{x},\mathbb{w})\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&=\sum_{\mathsf{aux}}\Pr[\mathsf{aux}]\cdot\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8\cdot\mathsf{q}'}\\&=\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\frac{\epsilon_{\mathsf{aux}}^{1.5}}{8\cdot\mathsf{q}'}\right]\\&\geq\frac{\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathsf{aux}}\right]^{1.5}}{8\cdot\mathsf{q}'} &(8)\\&=\frac{\epsilon^{1.5}}{8\cdot\mathsf{q}'} \ , &(9)\end{aligned}$$

where Equation (8) follows from Jensen's inequality and Equation (9) follows from the fact that $\mathop{\mathbb{E}}_{\mathsf{aux}}[\epsilon_{\mathsf{aux}}] = \epsilon$. Overall, since $\mathsf{q}' = \mathbf{K} \cdot \mathsf{q}$, we conclude that

$$\Pr\left[(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \geq \frac{\epsilon^{1.5}}{8\mathbf{K} \cdot \mathsf{q}} \ .$$

□

□

We now prove the main theorem of this section.

Proof of Theorem 6.2. Let $\tilde{\mathcal{P}}$ be a cheating prover and let $\epsilon_{\mathcal{D}}$ be the rogue soundness error of NARG with respect to $\mathcal{D}$ and Setup. Given Lemma 6.8 and the assumption that $\mathcal{R}$ is second-moment hard with respect to the distribution $\mathcal{D}$ and the setup algorithm Setup, it holds that either $\epsilon_{\mathcal{D}} \leq \frac{4\mathbf{K} \cdot \mathsf{q}}{2^\lambda}$ or,

$$\begin{aligned} \frac{\epsilon_{\mathcal{D}}^{1.5}}{8\mathbf{K} \cdot \mathsf{q}} &\leq \Pr\left[(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \\ &\leq \frac{\Delta \cdot \mathbb{E}\left[T^2_{A^*, \mathcal{D}}\right]}{|\mathcal{W}_\lambda|^\omega} \\ &\leq \frac{\Delta \cdot 4(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} \ . \end{aligned}$$

This leads to

$$\epsilon_{\mathcal{D}} \leq \left(\frac{\Delta \cdot 32\mathbf{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3} \ .$$

Overall, we derive the following

$$\begin{aligned} \epsilon_{\mathcal{D}} &\leq \max\left\{\left(\frac{\Delta \cdot 32\mathbf{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3}, \frac{4\mathbf{K} \cdot \mathsf{q}}{2^\lambda}\right\} \\ &\leq \left(\frac{\Delta \cdot 32\mathbf{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega}\right)^{2/3} + \frac{4\mathbf{K} \cdot \mathsf{q}}{2^\lambda} \ . \end{aligned}$$

□

7 Implications to the Schnorr and Okamoto schemes

In this section, we derive a concrete bound on the rogue-soundness error of the batch version of Schnorr and Okamoto schemes. We first show that each one is algebraic, then, assuming the second-moment assumption, we apply Corollary 8.9 and Theorem 6.2 in order to bound their rogue-instance security.

In the description of the schemes, we rely on the existence of a group generator algorithm $\mathsf{GroupGen}$ that on input 1^λ, outputs a description $(\mathbb{G}, p, g)$ of an abelian cyclic group $\mathbb{G}$ of prime order p, where g is a generator of the group. We consider the standard case that on input 1^λ, the algorithm $\mathsf{GroupGen}$ outputs a description of a group such that $p \geq 2^\lambda$. We denote by $t_{\mathsf{exp}} = t_{\mathsf{exp}}(\lambda)$ the time required for a single exponentiation if the group $\mathbb{G}$ where $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}$. For simplicity, we assume that sampling elements from $\mathbb{Z}_p$, arithmetic computations in $\mathbb{Z}_p$ and group operations in $\mathbb{G}$ can be done efficiently and are subsumed by t_{exp}.

7.1 The Schnorr protocol

We start by recalling the Schnorr identification scheme $\mathsf{ID}_{\mathsf{Schnorr}} = (\mathsf{Setup}, \mathsf{Gen}, \mathbf{P}_1, \mathbf{P}_2, \mathcal{C})$ which is defined as follows:

$\underline{\mathsf{Setup}(1^\lambda)}$: 1. $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ 2. Output $(\mathbb{G}, p, g)$ $\underline{\mathsf{Gen}(\mathsf{pp})}$: 1. Parse pp as $(\mathbb{G}, p, g)$ 2. $\mathbb{w} \leftarrow \mathbb{Z}_p$ 3. $\mathbb{x} = g^{\mathbb{w}}$ 4. Output $(\mathbb{x}, \mathbb{w})$ $\underline{\mathbf{V}(\mathsf{pp}, \mathbb{x}, \alpha, \beta, \gamma)}$: 1. Parse pp as $(\mathbb{G}, p, g)$ 2. If $g^\gamma = \alpha \cdot \mathbb{x}^\beta$ then output 1 and otherwise output 0	$\underline{\mathbf{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w})}$: 1. Parse pp as $(\mathbb{G}, p, g)$ 2. $r \leftarrow \mathbb{Z}_p$ 3. $\alpha = g^r$ 4. $\mathsf{st} = (\mathbb{w}, r)$ 5. Output (α, st) $\underline{\mathbf{P}_2(\mathsf{st}, \beta)}$: 1. Parse st as $(\mathbb{w}, r)$ 2. Output $\gamma = r + \beta \cdot \mathbb{w}$

Note that the scheme's challenge space $\mathcal{C}_{\mathsf{pp}}$ is $\mathbb{Z}_p$. One can observe that Schnorr identification scheme is algebraic.

Using construction Construction 5.3, we consider a batch version of Schnorr identification scheme as follows:

Construction 7.1. *Let* $(\mathsf{Setup}, \mathsf{F})$ *be a family of functions such that for every* $\mathsf{pp} = (\mathbb{G}, p, g)$ *produced by* $\mathsf{Setup}(1^\lambda)$ *the function* $\mathsf{F}_{\mathsf{pp}} \colon \mathbb{Z}_p \to \mathbb{G}$ *defined as follows:* $\mathsf{F}_{\mathsf{pp}}(x) = g^x$*. A batch version for Schnorr identification scheme* $\mathsf{ID}_{\mathsf{B\text{-}Schnorr}} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \mathcal{C})$ *is constructed as follows:*

$\underline{\mathsf{Setup}(1^\lambda)}$*:* 1. $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ 2. *Output* $(\mathbb{G}, p, g)$ $\underline{\mathsf{Gen}(\mathsf{pp})}$*:* 1. *Parse* pp *as* $(\mathbb{G}, p, g)$ 2. $\mathbb{w} \leftarrow \mathbb{Z}_p$ 3. $\mathbb{x} = \mathsf{F}_{\mathsf{pp}}(\mathbb{w})$ 4. *Output* $(\mathbb{x}, \mathbb{w})$ $\underline{\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta_1, \ldots, \beta_k, \gamma)}$*:* 1. *Parse* pp *as* $(\mathbb{G}, p, g)$ 2. *If* $\mathsf{F}_{\mathsf{pp}}(\gamma) = \alpha \cdot \Pi_{i=1}^{k} \mathbb{x}_i^{\beta_i}$ *then output* 1 *and otherwise output* 0	$\underline{\boldsymbol{P}_1(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k))}$*:* 1. *Parse* pp *as* $(\mathbb{G}, p, g)$ 2. *For every* $i \in [k]$*:* (a) *Sample* $r_i \leftarrow \mathbb{Z}_p$ (b) *Compute* $R_i = \mathsf{F}_{\mathsf{pp}}(r_i)$ 3. $\alpha = \Pi_{i=1}^{i=k} R_i$ 4. $\mathsf{st} = (\mathbb{w}_1, \ldots, \mathbb{w}_k, r_1, \ldots, r_k)$ 5. *Output* (α, st) $\underline{\boldsymbol{P}_2(\mathsf{st}, \beta_1, \ldots, \beta_k)}$*:* 1. *Parse* st *as* $(\mathbb{w}_1, \ldots, \mathbb{w}_k, r_1, \ldots, r_k)$ 2. *For every* $i \in [k]$*: compute* $\gamma_i = r_i + \beta_i \cdot \mathbb{w}_i$ 3. *Compute* $\gamma = \sum_{i=1}^{k} \gamma_i$ 4. *Output* γ

Note that the verifier performs $k+1$ exponentiation, therefore it runs in time $(k+1)t_{\mathsf{exp}}$. By Claim 5.5, Schnorr batch identification scheme has local special soundness. The extractor E performs only arithmetic operations in the ring $\mathbb{Z}_p$, and hence its running time is subsumed by t_{exp}. Additionally, Claim 5.17 shows if the Schnorr protocol has t_{Sim} zero-knowledge, then Schnorr batch Σ-protocol has $(\mathbf{K} \cdot t_{\mathsf{Sim}})$-time zero-knowledge. Note that Schnorr protocol has $2t_{\mathsf{exp}}$-time zero-knowledge, therefore, we combined with Corollary 8.9 we derive the following.

Theorem 7.2. *Let* $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), \mathsf{q} = \mathsf{q}(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$*. Let* $\mathsf{ID}_{\mathsf{B\text{-}Schnorr}} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be Schnorr batch identification scheme defined in (Construction 7.1). If the discrete-logarithm problem is second-moment hard with respect to* Gen*, then for any malicious prover* $\tilde{\boldsymbol{P}}$ *that runs in time* $t_{\tilde{\boldsymbol{P}}}$ *and issues* q *transcript-generation queries it holds that*

$$\Pr\left[\mathsf{StrongIdent}_{\mathsf{ID}_{\mathsf{B\text{-}Schnorr}}}(\tilde{\boldsymbol{P}}, \lambda)\right] \leq \left(\frac{32 \cdot (t_{\tilde{\boldsymbol{P}}} + 2\boldsymbol{K}(\mathsf{q}+1)t_{\mathsf{exp}})^2}{2^\lambda}\right)^{2/3} + \frac{4}{2^\lambda} \ .$$

Similarly, we consider the non-interactive batch argument of Schnorr protocol as constructed in Construction 6.1. Then ,we obtain the following theorem

Theorem 7.3. *Let* $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), \mathsf{q} = \mathsf{q}(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$*. Let* $\mathsf{NARG}_{\mathsf{B\text{-}Schnorr}} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *be the non-interactive batch argument constructed from Schnorr protocol as constructed is (Construction 6.1). If the discrete-logarithm problem is second-moment hard with respect to* $\mathsf{GroupGen}$*, then* $\mathsf{NARG}_{\mathsf{B\text{-}Schnorr}}$ *has* $(t_{\tilde{\mathcal{P}}}, \mathsf{q}, \epsilon)$*-rogue soundness error such that*

$$\epsilon(\lambda, t_{\tilde{\mathcal{P}}}, \mathsf{q}, \boldsymbol{K}) \leq \left(\frac{32\boldsymbol{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + (\boldsymbol{K}+1)t_{\mathsf{exp}})^2}{2^\lambda}\right)^{2/3} + \frac{4\boldsymbol{K} \cdot \mathsf{q}}{2^\lambda} \ .$$

7.2 The Okamoto protocol

We start by recalling the Okamoto identification scheme $\mathsf{ID}_{\mathsf{Okamoto}} = (\mathsf{Setup}, \mathsf{Gen}, \mathbf{P}_1, \mathbf{P}_2, \mathcal{C})$ which is defined as follows:

$\underline{\mathsf{Setup}(1^\lambda)}$: 1. $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ 2. $h \leftarrow \mathbb{G}$ 3. Output $((\mathbb{G}, p, g), h)$ $\underline{\mathsf{Gen}(\mathsf{pp})}$: 1. Parse pp as $((\mathbb{G}, p, g), h)$ 2. $w, w' \leftarrow \mathbb{Z}_p$ 3. $\mathbb{x} = g^w \cdot h^{w'}$ 4. $\mathbb{w} = (w, w')$ 5. Output $(\mathbb{x}, \mathbb{w})$ $\underline{\mathbf{V}(\mathsf{pp}, \mathbb{x}, \alpha, \beta, \gamma)}$: 1. Parse pp as $((\mathbb{G}, p, g), h)$ 2. Parse γ as (s, s') 3. If $g^s \cdot h^{s'} = \alpha \cdot \mathbb{x}^\beta$ then output 1 and otherwise output 0	$\underline{\mathbf{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w})}$: 1. Parse pp as $((\mathbb{G}, p, g), h)$ 2. $r, r' \leftarrow \mathbb{Z}_p$ 3. $\alpha = g^r \cdot h^{r'}$ 4. $\mathsf{st} = (\mathbb{w}, r, r')$ 5. Output (α, st) $\underline{\mathbf{P}_2(\mathsf{st}, \beta)}$: 1. Parse st as $(\mathbb{w}, r, r')$ 2. Parse $\mathbb{w}$ as (w, w') 3. $s = r + \beta \cdot w$ 4. $s' = r' + \beta \cdot w'$ 5. Output $\gamma = (s, s')$

Note that the scheme's challenge space $\mathcal{C}_{\mathsf{pp}}$ is $\mathbb{Z}_p$. One can simply observe that Okamoto identification scheme is algebraic identification scheme.

Using construction Construction 5.3, we consider a batch version of Okamoto identification scheme as follows:

Construction 7.4. *Let* $(\mathsf{Setup}, \mathsf{F})$ *be a family of one way functions such that for every* $\mathsf{pp} = ((\mathbb{G}, p, g), h)$ *produced by* $\mathsf{Setup}(1^\lambda)$ *the function* $\mathsf{F}_{\mathsf{pp}} \colon \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{G}$ *defined as follows:* $\mathsf{F}_{\mathsf{pp}}(x, x') = g^x \cdot h^{x'}$. *A batch version for Okamoto identification scheme* $\mathsf{ID}_{\mathsf{B\text{-}Okamoto}} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \mathcal{C})$ *is constructed as follows:*

$\underline{\mathsf{Setup}(1^\lambda)\text{:}}$

1. $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$
2. $h \leftarrow \mathbb{G}$
3. *Output* $((\mathbb{G}, p, g), h)$

$\underline{\mathsf{Gen}(\mathsf{pp})\text{:}}$

1. *Parse* pp *as* $((\mathbb{G}, p, g), h)$
2. $w, w' \leftarrow \mathbb{Z}_p$
3. $\mathbb{x} = g^w \cdot h^{w'}$
4. $\mathbb{w} = (w, w')$
5. *Output* $(\mathbb{x}, \mathbb{w})$

$\underline{\boldsymbol{V}(\mathsf{pp}, \mathbb{x}_1, \ldots, \mathbb{x}_k, \alpha, \beta_1, \ldots, \beta_k, \gamma)\text{:}}$

1. *Parse* pp *as* $((\mathbb{G}, p, g), h)$
2. *Parse* γ *as* (s, s')
3. *If* $\mathsf{F}_{\mathsf{pp}}(s, s') = \alpha \cdot \Pi_{i=1}^{k} \mathbb{x}_i^{\beta_i}$ *then output* 1 *and otherwise output* 0

$\underline{\boldsymbol{P}_1(\mathsf{pp}, (\mathbb{x}_1, \mathbb{w}_1), \ldots, (\mathbb{x}_k, \mathbb{w}_k))\text{:}}$

1. *Parse* pp *as* $((\mathbb{G}, p, g), h)$
2. *For every* $i \in [k]$*:*
   (a) *Sample* $r_i, r_i' \leftarrow \mathbb{Z}_p$
   (b) *Compute* $R_i = \mathsf{F}_{\mathsf{pp}}(r_i, r_i')$
3. $\alpha = \Pi_{i=1}^{i=k} R_i$
4. $\mathsf{st} = (\mathbb{w}_1, \ldots, \mathbb{w}_k, r_1, r_1', \ldots, r_k, r_k')$
5. *Output* (α, st)

$\underline{\boldsymbol{P}_2(\mathsf{st}, \beta_1, \ldots, \beta_k)\text{:}}$

1. *Parse* st *as* $(\mathbb{w}_1, \ldots, \mathbb{w}_k, r_1, r_1', \ldots, r_k, r_k')$
2. *For every* $i \in [k]$*:*
   (a) *Parse* $\mathbb{w}_i$ *as* (w_i, w_i')
   (b) *Compute* $s_i = r_i + \beta_i \cdot w_i$
   (c) *Compute* $s_i' = r_i' + \beta_i \cdot w_i'$
3. *Compute* $s = \sum_{i=1}^{k} s_i$
4. *Compute* $s' = \sum_{i=1}^{k} s_i'$
5. *Output* $\gamma = (s, s')$

Let $\mathcal{R}_2$ be the relation produced by Gen. The relation $\mathcal{R}_2$ consists of pairs $(g^x h^y, (x, y))$. Let $\mathcal{D}$ be a distribution over pairs (g^x, x). It is well known that the hardness of the relation $\mathcal{R}_2$ with respect to Gen is implied by the hardness of the discrete-logarithm relation with respect to $\mathcal{D}$. Therefore, if the discrete-logarithm relation is second-moment hard, then $\mathcal{R}_2$ is second-moment hard with $\omega = 1/2$ (Definition 3.2), since the witness space of $\mathcal{R}_2$ is of size p^2 compared to the witness space of discrete-logarithm which is of size p.

Note that the verifier performs $k + 1$ exponentiation, therefore its running time is $(k + 1) \cdot t_{\mathsf{exp}}$. By Claim 5.5, Okamoto batch identification scheme has local special soundness. The extractor E performs only arithmetic operations in the ring $\mathbb{Z}_p$, and hence its running time is subsumed by t_{exp}. Additionally, Claim 5.17 shows if the Okamoto protocol has t_{Sim} zero-knowledge, then Okamoto batch Σ-protocol has $(\mathbf{K} \cdot t_{\mathsf{Sim}})$-time zero-knowledge. Note that Okamoto protocol has t_{Sim}-time zero-knowledge such that $\mathsf{Sim} = 3 \cdot t_{\mathsf{exp}}$. Combined with Corollary 8.9 we derive the following.

Theorem 7.5. *Let* $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), \mathsf{q} = \mathsf{q}(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$*. Let* $\mathsf{ID}_{\mathsf{B\text{-}Okamoto}} = (\mathsf{Setup}, \mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be Okamoto batch identification scheme defined in (Construction 7.4). If the discrete-logarithm problem is second-moment hard with respect to* Gen*, then for any malicious prover* $\tilde{\boldsymbol{P}}$ *that runs in time* $t_{\tilde{\boldsymbol{P}}}$ *and issues* q *transcript-generation queries it holds that*

$$\Pr\left[\mathsf{StrongIdent}_{\mathsf{ID}_{\mathsf{B\text{-}Schnorr}}}(\tilde{\boldsymbol{P}}, \lambda)\right] \leq \left(\frac{32 \cdot (t_{\tilde{\boldsymbol{P}}} + 3\boldsymbol{K}(\mathsf{q}+1)t_{\mathsf{exp}})^2}{2^\lambda}\right)^{2/3} + \frac{4}{2^\lambda} \ .$$

Similarly, we consider the non-interactive batch argument of the Okamoto protocol as constructed in Construction 6.1. Then ,we obtain the following theorem

Theorem 7.6. *Let* $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), \mathsf{q} = \mathsf{q}(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$*. Let* $\mathsf{NARG}_{\mathsf{B\text{-}Okamoto}} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *be the non-interactive batch argument constructed from Okamoto protocol as constructed is (Construction 6.1). If the discrete-logarithm problem is second-moment hard with respect to* $\mathsf{GroupGen}$*, then* $\mathsf{NARG}_{\mathsf{B\text{-}Okamoto}}$ *has* $(t_{\tilde{\mathcal{P}}}, \mathsf{q}, \epsilon)$*-rogue soundness error such*

that

$$\epsilon(\lambda, t_{\tilde{\mathcal{P}}}, \mathsf{q}, \boldsymbol{K}) \leq \left(\frac{32\boldsymbol{K} \cdot \mathsf{q}(t_{\tilde{\mathcal{P}}} + (\boldsymbol{K}+1)t_{\mathsf{exp}})^2}{2^\lambda}\right)^{2/3} + \frac{4\boldsymbol{K} \cdot \mathsf{q}}{2^\lambda} \ .$$

8 Interactive proof of knowledge

In this section, we give a concrete bound on the rogue soundness error with respect to first-moment hard distribution of any batch Σ-protocol with t_W-time plus-one special soundness. Then, we extend this result for batch identification schemes which consist of a zero-knowledge batch Σ-protocol.

In Section 8.1, we construct a general algorithm for the collision game considered in Section 5.2. Then in Section 8.2, we give our main theorem of this section which shows the rogue-instance security with respect to a first-moment hard distribution of batch Σ-protocols.

8.1 General collision game

We now consider the collision game generalization from Section 5.2 and show the following lemma:

Lemma 8.1. *Let $H \in \{0,1\}^{R\times N}$ be a binary matrix and let ϵ be the fraction of 1-entries in H. Let $\mathcal{Q}_A$ be a random variable indicating the number of queries performed by A to H. Then, there exists an algorithm A with oracle access to H such that on input k the following holds:*

1. $\mathbb{E}[\mathcal{Q}_A] \leq k+1$
2. $\Pr[\mathsf{CollGame}_k(A,H)=1] \geq \epsilon - \frac{3k^2}{N}$

Proof. We consider the following algorithm A:

$A^H(k)$

1. Sample $\rho \leftarrow R$ and $\beta \leftarrow N$. If $H[\rho,\beta]=0$ abort.
2. Let $S, F = \{\beta\}$. While $|F| < k$:
   (a) Sample $\beta_i \leftarrow N \setminus S$ and set $S = S \cup \{\beta_i\}$.
   (b) If $H[\rho,\beta_i]=1$, set $F = F \cup \{\beta_i\}$.
   (c) If $|S| = N$, abort.
3. Return ρ and β_i for each $\beta_i \in F$.

We now prove each claim separately.

Claim 8.2. *It holds that $\mathbb{E}[\mathcal{Q}_A] \leq k+1$.*

Proof. Let ϵ_ρ be the fraction of 1-entries in row ρ in H. Note that either way, A performs at least 1 query. By the construction of A, assuming the first query to the matrix was 1-entry, finding another k entries with the value 1 can be modeled by a negative hypergeometric distribution (Definition 3.14). The population is of size $N-1$ and the number of "successes" in each row is $\epsilon_\rho(N-1)$. Thus, for every ρ, we consider two cases:

- Case 1: $\epsilon_\rho > \frac{1}{N}$. Following the negative hypergeometric distribution, the expected number of queries made to H is $\frac{k(N-1+1)}{\epsilon_\rho N-1+1} = \frac{k}{\epsilon_\rho}$.
- Case 2: $\epsilon_\rho \leq \frac{1}{N}$. In this case, the algorithm A performs at most $N-1$ queries which can be bounded by $\frac{k}{\epsilon_\rho}$.

Overall, the expected number of queries performed by A is

$$\mathbb{E}[\mathcal{Q}_\mathcal{A}] \leq 1 + \frac{1}{R}\cdot\sum_{\rho=0}^{R}\epsilon_\rho\cdot\frac{k}{\epsilon_\rho} = 1 + \frac{k}{R}\cdot\sum_{\rho=0}^{R}1 = k+1 \ .$$

□

It remains to show a lower bound on the probability that A succeeds in the collision game.

Claim 8.3. *It holds that* $\Pr[\mathsf{CollGame}_k(A,H)=1] \geq \epsilon - \frac{3k^2}{N}$.

Proof. We begin with a short high-level overview of the proof. Assuming the first query to the matrix was 1-entry, A continues to sample entries from the same row. Note that for each row, the number of 1-entries may be different which affects the success probability of the algorithm. Therefore, as in Claim 5.10, we "divide" the rows into "buckets" by the number of 1-entries in it.

Formally, for every $0 \leq d \leq N$, we let δ_d be the fraction of rows with exactly d 1-entries. If A sampled a row with at least $k+1$ entries with the value 1, then it necessarily finds $k+1$ different 1-entries and succeeds in $\mathsf{CollGame}_k(A,H)$. Therefore,

$$\Pr[\mathsf{CollGame}_k(A,H)=1] \geq \sum_{d=k+1}^{N} \delta_d \frac{d}{N} = \sum_{d=0}^{N} \delta_d \frac{d}{N} - \sum_{d=0}^{k+1} \delta_d \frac{d}{N} \ .$$

Note that $\sum_0^N \delta_d \frac{d}{N} = \epsilon$. Thus,

$$\Pr[\mathsf{CollGame}_k(A,H)=1] \geq \epsilon - \sum_{d=0}^{k+1} \frac{d}{N} \geq \epsilon - \frac{3k^2}{N} \ .$$

□

□

8.2 Rogue soundness error from the collision game

We now show how to invoke the algorithm for the collision game in order to extract a witness. Then, we prove the main theorem of this section which shows a bound on the rogue soundness error with respect to a first-moment hard distribution. Formally, we derive the following theorem:

Theorem 8.4. *Let* $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda, \boldsymbol{K}), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K})$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$. *Let* $\Pi = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a batch* Σ*-protocol with* t_W*-time plus-one special soundness for a relation* $\mathcal{R}$. *If* $\mathcal{R}$ *is first-moment hard with respect to a distribution* $\mathcal{D}$ *and the setup algorithm* Setup*, then* Π *has* $(t_{\tilde{\boldsymbol{P}}}, \epsilon)$*-rogue soundness error such that*

$$\epsilon_{\mathcal{D}}(\lambda, t_{\tilde{\boldsymbol{P}}}, t_{\mathcal{V}}, t_W, \boldsymbol{K}) \leq \frac{\Delta \cdot (\boldsymbol{K}+1) \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} + \frac{3\boldsymbol{K}^2}{|\mathcal{C}_{\mathsf{pp}}|} \ ,$$

where $t_{\boldsymbol{V}}$ *denotes the running time of the verifier* $\boldsymbol{V}$.

When referring to the non-interactive arguments constructed via the Fiat-Shamir paradigm [FS86], Theorem 8.4 directly derives the following corollary:

Corollary 8.5. *Let* $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\mathcal{P}}} = t_{\tilde{\mathcal{P}}}(\lambda, \boldsymbol{K}), t_{\mathcal{V}} = t_{\mathcal{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K})$ *be functions of the security parameter* $\lambda \in \mathbb{N}$ *and the bound on the number of instances* $\boldsymbol{K} \in \mathbb{N}$. *Let* $\Pi = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ *be a batch* Σ*-protocol with* t_W*-time plus-one special soundness for a relation* $\mathcal{R}$. *Let* $\mathsf{NARG} = (\mathcal{P}, \mathcal{V})$ *be the non-interactive batch argument of* Π *constructed via the Fiat-Shamir paradigm. If* $\mathcal{R}$ *is first-moment hard with respect to a distribution* $\mathcal{D}$ *and the setup algorithm* Setup*, then* Π *has* $(t_{\tilde{\mathcal{P}}}, \mathsf{q}, \epsilon_{\mathcal{D}})$*-rogue soundness error such that*

$$\epsilon_{\mathcal{D}}(\lambda, t_{\tilde{\mathcal{P}}}, t_{\mathcal{V}}, t_W, \mathsf{q}, \boldsymbol{K}) \leq \frac{\Delta \cdot \mathsf{q}(\boldsymbol{K}+1) \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\mathcal{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} + \frac{3\mathsf{q} \cdot \boldsymbol{K}^2}{|\mathcal{C}_{\mathsf{pp}}|} \ ,$$

where $t_{\mathcal{V}}$ *denotes the running time of the verifier* $\mathcal{V}$.

In order to prove Theorem 8.4 we first show the following lemma (which is interesting on its own):

Lemma 8.6. *Let $t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda, \boldsymbol{K}), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K})$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let $\Pi = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be a batch Σ-protocol with t_W-time plus-one special soundness for a relation $\mathcal{R}$. Let $t_{\boldsymbol{V}}$ denote the running time of the verifier $\boldsymbol{V}$ and let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ be a distribution over the relation where each $\mathcal{D}_\lambda$ produces $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_\lambda$. For every prover $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2)$ that runs in time $t_{\tilde{\boldsymbol{P}}}$, there exists an algorithm A^* such that:*

1. $\mathbb{E}\left[T_{A^*, \mathcal{D}_\lambda}\right] \leq (\boldsymbol{K}+1) \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)$.
2. $\Pr\left[(\mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda(\mathsf{pp}) \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \geq \epsilon - \frac{3\boldsymbol{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}$ *where ϵ is the rogue-soundness error of Π with respect to a distribution $\mathcal{D}$ and the setup algorithm* Setup.

Proof. We consider a binary matrix $H \in \{0,1\}^{R \times N}$, where the R rows correspond to $\tilde{\mathbf{P}}$'s randomness and the N columns correspond to $\mathbf{V}$'s randomness. Note that although $\tilde{\mathbf{P}}$'s and $\mathbf{V}$'s randomness depends on the number of instances that $\tilde{\mathbf{P}}$ outputs, we can always bound its size by the randomness size when $\tilde{\mathbf{P}}$ outputs $\mathbf{K}$ instances.

An entry of H equals 1 if and only if the corresponding transcript (between $\tilde{\mathbf{P}}$ and $\mathbf{V}$) is accepting. Recall that every algorithm A for the collision game on input k, aims to find $k+1$ entries with the value 1 in the same row. As $\tilde{\mathbf{P}}$'s randomness is fixed along one row, finding $k+1$ entries with the value 1 in the same row corresponds to finding $k+1$ accepting transcripts with a common first message and pairwise distinct challenges, which by the plus-one special soundness allows extracting k witnesses.

Let A be the algorithm for the collision game constructed in Lemma 8.1, we construct the algorithm A^* as follows:

$A^*(\mathsf{pp}, \mathbb{x}_1)$

1. Initialize an empty mapping M between the randomness used by $\tilde{\mathbf{P}}$ and $\mathbf{V}$ and transcript between them.
2. Invoke $A(\mathbf{K})$. When A performs a query on (ρ, β) answer as follows:
   (a) Invoke $((\tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k), \alpha, \mathsf{st}) \leftarrow \tilde{\mathbf{P}}_1(1^\lambda, \mathsf{pp}, \mathbb{x}_1; \rho)$.
   (b) Invoke $\gamma \leftarrow \tilde{\mathbf{P}}_2(\beta, \mathsf{st})$.
   (c) Set $M[(\rho, \beta)] \leftarrow (\alpha, \beta, \gamma)$.
   (d) Return $\mathbf{V}(\mathsf{pp}, \mathbb{x}_1, \tilde{\mathbb{x}}_2, \ldots, \tilde{\mathbb{x}}_k, \alpha, \beta, \gamma)$ as the answer to the query.
3. When A outputs $\rho, \beta_1, \ldots, \beta_{\mathbf{K}+1}$: for every $j \in [\mathbf{K}+1]$: set $(\alpha_j^*, \beta_j^*, \gamma_j^*) \leftarrow M[\rho, \beta_j]$.
4. Let k^* be the number of instances $\tilde{\mathbf{P}}_1$ outputs when running with the randomness ρ. Run $\tilde{\mathbb{w}}_1, \ldots, \tilde{\mathbb{w}}_{k^*} \leftarrow \Pi.\mathsf{E}(\alpha_1^*, (\beta_1^*, \gamma_1^*), \ldots, (\beta_{k^*+1}^*, \gamma_{k^*+1}^*))$.
5. Output $\tilde{w}_1$.

We prove each claim separately.

Claim 8.7. *It holds that $\mathbb{E}\left[T_{A^*, \mathcal{D}_\lambda}\right] \leq (\boldsymbol{K}+1) \cdot (t_{\tilde{\boldsymbol{P}}} + t_{\boldsymbol{V}} + t_W)$.*

Proof. Observe that whenever A query H, A^* invokes $\tilde{\mathbf{P}}$ and $\mathbf{V}$. Thus, the expected number of invocations that A^* performs to $\tilde{\mathbf{P}}$ and $\mathbf{V}$ is the expected number of queries performed by A. Thus,

$$\mathbb{E}\left[T_{A^*, \mathcal{D}_\lambda}\right] \leq \mathbb{E}\left[\mathcal{Q}_A\right] \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}}) + t_W \leq (\mathbf{K}+1) \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W) \ .$$

□

Claim 8.8. *It holds that* $\Pr\left[(\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\boldsymbol{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\geq\epsilon-\frac{3\boldsymbol{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}$ *where* ϵ *is the rogue-soundness error of* Π *with respect to a distribution* $\mathcal{D}$ *and the setup algorithm* Setup.

Proof. Whenever A succeeds in the collision game with H, the algorithm A^* outputs a witness for $\mathbb{x}_1$. Thus,

$$\begin{aligned}&\Pr\left[(\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&=\Pr\left[\mathsf{CollGame}_{\mathbf{K}}(A,H)=1\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\end{array}\right].\end{aligned}$$

Let aux denote that variable for a tuple $(\mathsf{pp},\mathbb{x})$. For every $\mathsf{aux}=(\mathsf{pp},\mathbb{x})$, we let

$$\epsilon_{\mathsf{aux}}=\Pr\left[\begin{array}{l}\mathbf{V}(\mathsf{pp},\mathbb{x},\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k,\alpha,\beta,\gamma)=1\\\text{conditioned on }\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x},\mathbb{w})\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\end{array}\;\middle|\;\begin{array}{l}((\tilde{\mathbb{x}}_2,\ldots,\tilde{\mathbb{x}}_k),\alpha,\mathsf{st})\leftarrow\tilde{\mathbf{P}}_1(\mathsf{pp},\mathbb{x})\\\beta\leftarrow\mathcal{C}_{\mathsf{pp}}\\\gamma\leftarrow\tilde{\mathbf{P}}_2(\mathsf{st},\beta)\end{array}\right].$$

Conditioned on $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})$ and $(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda$, the collision game matrix has ϵ_{aux} fraction of 1-entries for $\mathsf{aux}=(\mathsf{pp},\mathbb{x}_1)$. Thus, conditioned on $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})$ and $(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda$, the probability that A succeeds in the collision game is $\epsilon_{\mathsf{aux}}-\frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}$. Therefore,

$$\begin{aligned}&\Pr\left[(\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\\&=\sum_{\mathsf{aux}}\Pr\left[\begin{array}{l}\mathsf{CollGame}_{\mathbf{K}}(A,H)=1\\\text{conditioned on }\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\end{array}\right]\cdot\Pr\left[\mathsf{aux}\right]\\&=\sum_{\mathsf{aux}}\left(\epsilon_{\mathsf{aux}}-\frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}\right)\cdot\Pr\left[\mathsf{aux}\right]\\&=\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathsf{aux}}-\frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}\right]\\&=\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathbb{x}_1}\right]-\frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}\;.\end{aligned}$$

Note that $\mathop{\mathbb{E}}_{\mathsf{aux}}\left[\epsilon_{\mathsf{aux}}\right]=\epsilon$, and thus,

$$\Pr\left[(\mathbb{x}_1,\tilde{\mathbb{w}}_1)\in\mathcal{R}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathbf{K})\\(\mathbb{x}_1,\mathbb{w}_1)\leftarrow\mathcal{D}_\lambda(\mathsf{pp})\\\tilde{\mathbb{w}}_1\leftarrow A^*(\mathsf{pp},\mathbb{x}_1)\end{array}\right]\geq\epsilon-\frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|}\;.$$

□

□

We are now ready to show a bound on the rogue soundness error of batch Σ-protocol with plus-one special soundness.

Proof of Theorem 8.4. Let $\tilde{\mathbf{P}}$ be a cheating prover and let $\epsilon_{\mathcal{D}}$ be the rogue soundness error with respect to $\mathcal{D}$. Given Lemma 8.6 and the assumption that $\mathcal{R}$ is first-moment hard with respect to the distribution $\mathcal{D}$ and the setup algorithm Setup, it holds that

$$\begin{aligned}\epsilon - \frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|} &\leq \Pr\left[(\mathbb{x}_1, \tilde{\mathbb{w}}_1) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(\mathbf{K}) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{D}_\lambda \\ \tilde{\mathbb{w}}_1 \leftarrow A^*(\mathsf{pp}, \mathbb{x}_1) \end{array}\right] \\ &\leq \frac{\Delta \cdot \mathbb{E}\left[T_{A^*,\mathcal{D}}\right]}{|\mathcal{W}_\lambda|^\omega} \\ &\leq \frac{\Delta \cdot (\mathbf{K}+1)(t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} \ .\end{aligned}$$

This leads to

$$\epsilon \leq \frac{\Delta \cdot (\mathbf{K}+1) \cdot (t_{\tilde{\mathbf{P}}} + t_{\mathbf{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} + \frac{3\mathbf{K}^2}{|\mathcal{C}_{\mathsf{pp}}|} \ .$$

□

8.3 Batch identification schemes

Recall that a batch identification scheme ID consists of a batch Σ-protocol Π such that the execution of ID is as the execution of Π where each public key pk corresponds to an instance and a secret key sk correspond to a witness. Thus, as discussed in Section 5.4, if Π has t_{Sim}-time zero-knowledge, any malicious prover that runs in time $t_{\tilde{\mathbf{P}}}$ and performs q queries to $\mathsf{Trans}_{\mathsf{pk}_1,\mathsf{sk}_1}(\cdot)$, can be simulated by a malicious prover that runs in time $t_{\tilde{\mathbf{P}}} + \mathsf{q} \cdot t_{\mathsf{Sim}}$. Thus, from Theorem 8.4 we derive the following corollary:

Corollary 8.9. *Let $\Delta = \Delta(\lambda), \omega = \omega(\lambda), t_{\tilde{\boldsymbol{P}}} = t_{\tilde{\boldsymbol{P}}}(\lambda), t_{\boldsymbol{V}} = t_{\boldsymbol{V}}(\lambda, \boldsymbol{K}), t_W = t_W(\lambda, \boldsymbol{K}), t_{\mathsf{Sim}} = t_{\mathsf{Sim}}(\lambda, \boldsymbol{K}), \mathsf{q} = \mathsf{q}(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$ and the bound on the number of instances $\boldsymbol{K} \in \mathbb{N}$. Let $\Pi = (\mathsf{Gen}, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{V}, \mathcal{C})$ be a batch identification scheme with t_W-time plus-one special soundness and t_{Sim}-time zero-knowledge for a relation $\mathcal{R}$. If $\mathcal{R}$ is first-moment hard with respect to Gen, then for any malicious prover $\tilde{\boldsymbol{P}}$ that runs in time $t_{\tilde{\boldsymbol{P}}}$ and issues q transcript-generation queries it holds that*

$$\Pr\left[\mathsf{StrongIdent}_{\tilde{\boldsymbol{P}},\Pi}(\lambda)\right] \leq \frac{\Delta \cdot (\boldsymbol{K}+1) \cdot (t_{\tilde{\boldsymbol{P}}} + \mathsf{q} \cdot t_{\mathsf{Sim}} + t_{\boldsymbol{V}} + t_W)}{|\mathcal{W}_\lambda|^\omega} + \frac{3\boldsymbol{K}^2}{|\mathcal{C}_{\mathsf{pp}}|} \ ,$$

where $t_{\boldsymbol{V}}$ denotes the running time of the verifier $\boldsymbol{V}$.

9 Proving expected-time hardness in generic models

In this section, we present a generic framework for analyzing expected-time hardness of cryptographic problems. In fact, applying our framework proves the second-moment assumption (Definition 3.2) for the discrete-logarithm problem in the generic group model. Shoup [Sho97] analyzed generic hardness of the discrete-logarithm problem with respect to strict time algorithms. He showed that any generic t-time algorithm that solves the discrete-logarithm problem has success probability at most $\epsilon \leq t^2/p$. Applying our framework yields a bound of $\epsilon \leq \mathbb{E}\left[T_A^2\right]/p$ when considering *unbounded* algorithms where T_A denotes the random variable indicating the algorithm's running time.

Our framework is inspired by [JT20] which showed a generic framework to prove bounds with respect to expected-time algorithms when considering only the first-moment of the expected running time. Their result proves the first-moment assumption (Definition 3.1) but cannot be used to derive the second-moment assumption.

In Section 9.1 we introduce our framework for proving expected-time hardness. In Section 9.3 we derive the second-moment assumption for the discrete-logarithm problem by applying our framework in the generic group model. In Section 9.2 and Section 9.4 we show the expected-time hardness of the collision-resistance of a random-oracle and of SNARKs.

9.1 Our framework

Definition 9.1 (Monotonic predicate)**.** *A predicate P is* monotonic *if for every tr such that $P(\mathsf{tr}) = 1$, it holds that $P(\mathsf{tr}||\mathsf{tr}') = 1$ for every tr'.*

We consider distributions $\mathcal{D}(\lambda)$ which produces an oracle $\mathcal{O}$ and define the hardness of a predicate as follows:

Definition 9.2 (Hard predicate)**.** *A predicate P is* ϵ-hard *if for every strict time algorithm $\mathcal{A}_t$ it holds that*

$$\Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}_t{}^{\mathcal{O}}(\mathsf{in}) \end{array}\right] \leq \epsilon(t) \ .$$

In addition, we define history-oblivious predicates. Intuitively, this family of predicates includes predicates on which each query is oblivious to the history of the query-answer list (see Section 2.6 for further discussion). We define history-oblivious by considering the hardness to set the predicate to output 1 on input $\mathsf{tr}\|(x,y)$ where (x,y) is a fresh query-answer pair and tr is a query-answer list on which the predicate outputs 0.

For any list of query-answer pairs μ we denote by $\mathcal{D}(\lambda,\mu)$ the distribution $\mathcal{D}(\lambda)$ of all oracles such that for every $(x_i,y_i) \in \mu$ it holds that $y_i = \mathcal{O}(x_i)$. We let X, Y be the query and answer spaces.

Definition 9.3 (History-oblivious predicate)**.** *Let P be an ϵ-hard predicate. We say that P is* history-oblivious *with respect to $\mathcal{O}$ if there is a function $\kappa(\cdot)$, such that for every $t \in \mathbb{N}$ the following holds:*

1. *For every $0 \leq i \leq t$, every trace tr of length i with $P(\mathsf{tr}) = 0$, and any query $x \in X$:*

$$\Pr\left[P(\mathsf{tr}\|(x,y)) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda,\mathsf{tr}) \\ y = \mathcal{O}(x) \end{array}\right] \leq \kappa(i) \ .$$

2. $\sum_{j=0}^{t} \kappa(j) \leq \epsilon(t)$.

(Above, the length of a trace is the number of query/answer pairs it contains.) We consider experiments relative to an oracle, for which their security relies on the trace between the adversary and the oracle. We capture this using a monotonic predicate on the trace. Formally, we define the following:

Definition 9.4 (δ-bounded experiment)**.** *Let $\mathsf{Exp}^{\mathcal{O}}$ be an experiment with oracle access $\mathcal{O}$, and let $\delta = \delta(\lambda)$ be a function of the security parameter $\lambda \in \mathbb{N}$. We say that $\mathsf{Exp}^{\mathcal{O}}$ is* δ-bounded *with respect to a monotonic predicate P if for every (bounded and unbounded) algorithm $\mathcal{A}$ it holds that,*

$$\Pr\left[\mathsf{Exp}^{\mathcal{O}}(\mathsf{in},\mathsf{out}) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{O}}(\mathsf{in}) \end{array}\right] \leq \Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{O}}(\mathsf{in}) \end{array}\right] + \delta \ .$$

Given the definitions above, we prove the following theorem.

Theorem 9.5. *Let $\mathsf{Exp}^{\mathcal{O}}$ be a δ-bounded experiment with respect to a predicate P which is ϵ-hard. If P is history-oblivious, then, for every* unbounded *algorithm $\mathcal{A}$ it holds that,*

$$\Pr\left[\mathsf{Exp}^{\mathcal{O}}(\mathsf{in},\mathsf{out})=1 \;\middle|\; \begin{array}{l}\mathcal{O}\leftarrow\mathcal{D}(\lambda)\\ \mathsf{out}\overset{\mathsf{tr}}{\leftarrow}\mathcal{A}^{\mathcal{O}}(\mathsf{in})\end{array}\right]\leq\mathbb{E}\left[\epsilon(t)\right]+\delta\ .$$

In particular, Theorem 9.5 allows us to recover the same bounds given in [JT20], which is captured in the following corollary.

Corollary 9.6. *Let $\mathsf{Exp}^{\mathcal{O}}$ be a δ-bounded experiment with respect to a predicate P which is ϵ-hard where $\epsilon(t)=\frac{\Delta t^d}{N}$ for $\Delta,d,N\geq 1$. If P is history-oblivious, then, for every* unbounded *algorithm $\mathcal{A}$ it holds that,*

$$\Pr\left[\mathsf{Exp}^{\mathcal{O}}(\mathsf{in},\mathsf{out})=1 \;\middle|\; \begin{array}{l}\mathcal{O}\leftarrow\mathcal{D}(\lambda)\\ \mathsf{out}\overset{\mathsf{tr}}{\leftarrow}\mathcal{A}^{\mathcal{O}}(\mathsf{in})\end{array}\right]\leq\sqrt[d]{\epsilon(\mathbb{E}\left[T_{\mathcal{A}}\right])}+\delta=\sqrt[d]{\frac{\Delta}{N}}\cdot\mathbb{E}\left[T_{\mathcal{A}}\right]+\delta\ ,$$

where $T_{\mathcal{A}}$ is a random variable indicating the number of queries performed by $\mathcal{A}$ until he stops, when given access to an oracle $\mathcal{O}$.

First, we prove Theorem 9.5 and then after we prove Corollary 9.6.

Proof of Theorem 9.5. Let tr_i be the first i pairs in the query-answer list between the algorithm and the oracle $\mathcal{O}$. Let Y_i be an indicator random variable for the event that

1. $|\mathsf{tr}|\geq i$; and
2. $P(\mathsf{tr}_i)=1$; and
3. $P(\mathsf{tr}_{i-1})=0$.

Note that, the events $Y_i=1$ are mutually exclusive, thus:

$$\begin{aligned}&\Pr\left[P(\mathsf{tr})=1 \;\middle|\; \begin{array}{l}\mathcal{O}\leftarrow\mathcal{D}(\lambda)\\ \mathsf{out}\overset{\mathsf{tr}}{\leftarrow}\mathcal{A}^{\mathcal{O}}(\mathsf{in})\end{array}\right]\\ &=\sum_{i=1}^{\infty}\Pr\left[Y_i=1 \;\middle|\; \begin{array}{l}\mathcal{O}\leftarrow\mathcal{D}(\lambda)\\ \mathsf{out}\overset{\mathsf{tr}}{\leftarrow}\mathcal{A}^{\mathcal{O}}(\mathsf{in})\end{array}\right]\ ,\end{aligned}$$

To simplify the notation throughout the proof, we omit the explicit reference to the probability taken over the sampling of the oracle $\mathcal{O}\leftarrow\mathcal{D}(\lambda)$ and the execution of the algorithm.

Let $T_{\mathcal{A}}=T_{\mathcal{A}}(\lambda)$ be a random variable indicating the number of queries performed by $\mathcal{A}$ until he stops, when given access to an oracle $\mathcal{O}$. Note that for every $i\in\mathbb{N}$, it holds that $Y_i=1$ only if the number of queries performed by the algorithm is at least i. Thus,

$$\begin{aligned}&\Pr\left[P(\mathsf{tr})=1 \;\middle|\; \begin{array}{l}\mathcal{O}\leftarrow\mathcal{D}(\lambda)\\ \mathsf{out}\overset{\mathsf{tr}}{\leftarrow}\mathcal{A}^{\mathcal{O}}(\mathsf{in})\end{array}\right]\\ &=\sum_{i=1}^{\infty}\Pr\left[Y_i=1 \;\middle|\; T_{\mathcal{A}}\geq i\ \right]\cdot\Pr[T_{\mathcal{A}}\geq i]\\ &\leq\sum_{i=1}^{\infty}\Pr\left[Y_i=1 \;\middle|\; T_{\mathcal{A}}\geq i\ \right]\cdot\sum_{t=i}^{\infty}\Pr[T_{\mathcal{A}}=t]\end{aligned}$$

The following claim (which we prove later on) shows an upper bound on the above term $\Pr\left[Y_i=1 \;\middle|\; T_{\mathcal{A}}\geq i\ \right]$.

Claim 9.7. *If P is ϵ-hard and history-oblivious, for every $i \in \mathbb{N}$ it holds that*

$$\Pr\left[Y_i = 1 \;\middle|\; T_{\mathcal{A}} \geq i \;\right] \leq \kappa(i) \;\; .$$

Given Claim 9.7 it holds that,

$$\begin{aligned}
&\Pr\left[P\left(\mathsf{tr}\right) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{O}}\left(\mathsf{in}\right) \end{array}\right] \\
&\leq \sum_{i=1}^{\infty} \kappa(i) \cdot \sum_{t=i}^{\infty} \Pr\left[T_{\mathcal{A}} = t\right] \\
&= \sum_{t=1}^{\infty} \Pr\left[T_{\mathcal{A}} = t\right] \cdot \sum_{i=1}^{t} \kappa(i) \qquad (10) \\
&\leq \sum_{t=1}^{\infty} \Pr\left[T_{\mathcal{A}} = t\right] \cdot \epsilon(t) \qquad (11) \\
&= \mathbb{E}\left[\epsilon(t)\right] \;\; ,
\end{aligned}$$

where Equation (10) is rearranging the summation and Equation (11) follows from the fact that P is ϵ-hard and history-oblivious. Overall, we conclude that,

$$\Pr\left[\mathsf{Exp}^{\mathcal{O}}(\mathsf{in}, \mathsf{out}) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{O}}\left(\mathsf{in}\right) \end{array}\right] \leq \mathbb{E}\left[\epsilon(t)\right] + \delta \;\; .$$

For completeness, we now prove Claim 9.7.

Proof of Claim 9.7. For every i it holds that,

$$\begin{aligned}
&\Pr\left[Y_i = 1 \;\middle|\; T_{\mathcal{A}} \geq i \;\right] \\
&= \Pr\left[P\left(\mathsf{tr}_i\right) = 1 \;\wedge\; P\left(\mathsf{tr}_{i-1}\right) = 0 \;\middle|\; T_{\mathcal{A}} \geq i \;\right] \\
&\leq \Pr\left[P\left(\mathsf{tr}_i\right) = 1 \;\middle|\; \begin{array}{l} T_{\mathcal{A}} \geq i \\ P\left(\mathsf{tr}_{i-1}\right) = 0 \end{array}\right] \;\; .
\end{aligned}$$

Let (x, y) be the i-th query-answer pair in tr_i. Since the events $T_{\mathcal{A}} \geq i$ and $P\left(\mathsf{tr}_{i-1}\right) = 0$ depends only on tr_{i-1}, we can find a query-answer list tr'_{i-1} such that $P\left(\mathsf{tr}'_{i-1}\right) = 0$ and leads to $T_{\mathcal{A}} \geq i$. Thus, if

$$\Pr\left[P\left(\mathsf{tr}_i\right) = 1 \;\middle|\; \begin{array}{l} T_{\mathcal{A}} \geq i \\ P\left(\mathsf{tr}_{i-1}\right) = 0 \end{array}\right] > \kappa(i) \;\; ,$$

then, there exists a query-answer list tr'_{i-1} such that

$$\Pr\left[P\left(\mathsf{tr}'_{i-1} \| (x, y)\right) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda, \mathsf{tr}'_{i-1}) \\ y = \mathcal{O}(x) \end{array}\right] > \kappa(i) \;\; ,$$

in contradiction to the fact that P is history-oblivious. □

□

We next give a proof of Corollary 9.6.

Proof of Corollary 9.6. We show that if $\epsilon(t) = \frac{\Delta t^d}{N}$, then $\mathbb{E}[\epsilon(t)] \leq \sqrt[d]{\epsilon(\mathbb{E}[T_{\mathcal{A}}])}$. By the definition of $\epsilon(t)$, it holds $\epsilon(t) \leq 1$ for every t. Therefore, when we consider $\epsilon(t) = \frac{\Delta t^d}{N}$ we implicitly consider $\epsilon(t) = \min\{1, \frac{\Delta t^d}{N}\}$. Thus,

$$\begin{aligned}
\mathbb{E}[\epsilon(t)] &= \mathbb{E}\left[\min\left\{1, \frac{\Delta t^d}{N}\right\}\right] \\
&= \sum_{t=0}^{\sqrt[d]{N/\Delta}} \frac{\Delta t^d}{N} \cdot \Pr[T_{\mathcal{A}} = t] + \sum_{t=\sqrt[d]{N/\Delta}+1}^{\infty} \Pr[T_{\mathcal{A}} = t] \\
&\leq \sum_{t=0}^{\sqrt[d]{N/\Delta}} \left(\frac{\Delta t^d}{N}\right)^{1/d} \cdot \Pr[T_{\mathcal{A}} = t] + \sum_{t=\sqrt[d]{N/\Delta}+1}^{\infty} \sqrt[d]{\frac{\Delta}{N}}\, t \cdot \Pr[T_{\mathcal{A}} = t] \qquad (12) \\
&= \sum_{t=0}^{\infty} \sqrt[d]{\frac{\Delta}{N}}\, t \cdot \Pr[T_{\mathcal{A}} = t] \\
&= \sqrt[d]{\frac{\Delta}{N}} \cdot \mathbb{E}[T_{\mathcal{A}}] \\
&= \sqrt[d]{\frac{\Delta}{N} \cdot \mathbb{E}[T_{\mathcal{A}}]^d} \\
&= \sqrt[d]{\epsilon(\mathbb{E}[T_{\mathcal{A}}])}\ ,
\end{aligned}$$

where Equation (12) holds since when $t \leq \sqrt[d]{N/\Delta}$, it holds that $\frac{\Delta t^d}{N} \leq 1$, and when $t \geq \sqrt[d]{N/\Delta}$ it holds that $\sqrt[d]{\frac{\Delta}{N}} t \geq 1$. Therefore, combined with Theorem 9.5, if $\epsilon(t) = \frac{\Delta t^d}{N}$ we conclude that

$$\Pr\left[\mathsf{Exp}^{\mathcal{O}}(\mathsf{in}, \mathsf{out}) = 1 \;\middle|\; \begin{array}{l} \mathcal{O} \leftarrow \mathcal{D}(\lambda) \\ \mathsf{out} \xleftarrow{\mathsf{tr}} \mathcal{A}^{\mathcal{O}}(\mathsf{in}) \end{array}\right] \leq \mathbb{E}[\epsilon(t)] + \delta \leq \sqrt[d]{\epsilon(\mathbb{E}[T_{\mathcal{A}}])} + \delta = \sqrt[d]{\frac{\Delta}{N}} \cdot \mathbb{E}[T_{\mathcal{A}}] + \delta\ .$$

□

9.2 Collision-resistance of a random oracle.

In this section, we show the expected-time hardness of the collision-resistance of a random oracle. Here, an algorithm is given oracle access to a random oracle $f \in \mathcal{U}(\lambda)$ and aims to find $x \neq x'$ such that $f(x) = f(x')$. Formally,

Lemma 9.8. *For any query algorithm A, let $T_A = T_A(\lambda)$ be a random variable indicating the number of queries performed by A until he stops, when given access to a random oracle f that outputs λ bits. Then, for every algorithm A it holds that*

$$\Pr\left[\begin{array}{l} f(x) = f(x') \\ \wedge\ x \neq x' \end{array} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \leftarrow A^f \end{array}\right] \leq \frac{1}{2} \cdot \frac{\mathbb{E}[T_A^2]}{2^\lambda} + \frac{1}{2^\lambda}\ .$$

Proof. Let tr be the query-answer list between the algorithm A and the random oracle f and we let P be the predicate as follows:

$$P(\mathsf{tr}) = \begin{cases} 1\ \exists (x, y), (x', y') \in \mathsf{tr} \text{ s.t } x \neq x' \text{ and } y = y' \\ 0 \text{ otherwise} \end{cases}\ .$$

We consider two cases:

- Case 1: A outputs $x, x' \in \mathsf{tr}$. If the algorithm finds a collision, then there exists a query on which the predicate outputs 1. Thus,

$$\Pr\left[\begin{matrix} f(x) = f(x') \\ \wedge\, x \neq x' \end{matrix} \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \leftarrow A^f \end{matrix}\right] \leq \Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \xleftarrow{\mathsf{tr}} A^f \end{matrix}\right] .$$

- Case 2: A outputs $x, x' \notin \mathsf{tr}$. Then, the algorithm finds a collision with probability $1/2^\lambda$.

We construct the experiment $\mathsf{RO\text{-}Coll}^f$ as follows:

$\mathsf{RO\text{-}Coll}^f(x, x')$:

1. Return 1 if and only if $x \neq x'$ and $f(x) = f(x')$.

Note that

$$\begin{aligned}\Pr\left[\begin{matrix} f(x) = f(x') \\ \wedge\, x \neq x' \end{matrix} \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \leftarrow A^f \end{matrix}\right] &= \Pr\left[\mathsf{RO\text{-}Coll}^f(x, x') = 1 \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \leftarrow A^f \end{matrix}\right] \\ &\leq \Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \xleftarrow{\mathsf{tr}} A^f \end{matrix}\right] + \frac{1}{2^\lambda} .\end{aligned}$$

Note that P is a monotonic predicate, therefore, $\mathsf{RO\text{-}Coll}^f$ is $\frac{1}{2^\lambda}$-bounded with respect to the predicate P (Definition 9.4). In order to use Theorem 9.5, it remains to show that P is ϵ-hard and history-oblivious. Recall that the definition of ϵ-hard predicates considers strict time algorithms. Let A_t be a t-query algorithm and let Y_i denote the event in which A_t finds the first collision in the i-th query. Overall,

$$\Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \xleftarrow{\mathsf{tr}} A^f \end{matrix}\right] = \Pr\left[\exists i : Y_i\right] \leq \sum_1^t \Pr\left[Y_i\right] = \sum_1^t \frac{i-1}{2^\lambda} \leq \frac{1}{2} \cdot \frac{t^2}{2^\lambda} .$$

Therefore, P is ϵ-hard where $\epsilon(t) = \frac{1}{2} \cdot \frac{t^2}{2^\lambda}$. We define $\kappa(i) = \frac{i-1}{2^\lambda}$, and we get that for any trace of length i (which for the predicate outputs 0), and any query x it holds that

$$\Pr\left[P\left(\mathsf{tr} \| (x, y)\right) = 1 \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ y = f(x) \end{matrix}\right] \leq \kappa(i) .$$

As we have shown above, it holds that $\sum_1^t \kappa(i) \leq \epsilon(t)$. Thus, P is history-oblivious.

By applying Theorem 9.5, it holds that

$$\Pr\left[\begin{matrix} f(x) = f(x') \\ \wedge\, x \neq x' \end{matrix} \;\middle|\; \begin{matrix} f \leftarrow \mathcal{U}(\lambda) \\ (x, x') \leftarrow A^f \end{matrix}\right] \leq \mathbb{E}\left[\epsilon(t)\right] + \frac{1}{2^\lambda} = \frac{1}{2} \cdot \frac{\mathbb{E}\left[T_A^2\right]}{2^\lambda} + \frac{1}{2^\lambda} .$$

□

9.3 Expected-time hardness of DL in the GGM

The generic group model Let p be a prime and let $\sigma \colon \mathbb{Z}_p \to \{0,1\}^s$ be a random injective mapping such that $s \geq \log p$. A generic algorithm takes as input $\sigma(x_1), \ldots, \sigma(x_k)$ and given an oracle access to a group operation oracle $\mathcal{O}$ that satisfies the following:

$$\mathcal{O}(\sigma(a), \sigma(b), \pm) = \sigma(a \pm b) .$$

Lemma 9.9. *For any query algorithm A, let $T_A = T_A(\lambda)$ be a random variable indicating the number of queries performed by A until he stops. Then, for every generic algorithm A it holds that*

$$\Pr\left[x' = x \;\middle|\; x' \leftarrow A^{\mathcal{O}}(p, \sigma(1), \sigma(x))\right] \leq \frac{1}{2} \cdot \frac{\mathbb{E}\left[T_A^2\right]}{p} + \frac{1}{p} \ ,$$

where the probability is taken over the random choice of σ and $x \in \mathbb{Z}_p$.

Proof. In the analysis of the discrete-logarithm problem in the generic group model, it suffices to focus on the probability of finding a collision in the group operation oracle. This is because if an algorithm can find a collision, it can exploit this to solve the discrete-logarithm problem efficiently. Thus, we consider the following predicate:

$$P(\mathsf{tr}) = \begin{cases} 1 \ \exists (x,y),(x',y') \in \mathsf{tr} \text{ s.t } x \neq x' \text{ and } y = y' \\ 0 \text{ otherwise} \end{cases} \ ,$$

where tr is the query-answer list between the algorithm A and the oracle $\mathcal{O}$. We construct the experiment $\mathsf{DL\text{-}GGM}$ as follows:

$\mathsf{DL\text{-}GGM}^{\mathcal{O}}((p, \sigma(1), \sigma(x)), x')$:

1. Compute $\sigma(x')$ using $\sigma(1)$ and queries to $\mathcal{O}$ (e.g, using repeated squaring).
2. Return 1 if and only if $\sigma(x) = \sigma(x')$.

Note that

$$\begin{aligned} \Pr\left[x' = x \;\middle|\; x' \leftarrow A^{\mathcal{O}}(p, \sigma(1), \sigma(x))\right] &= \Pr\left[\mathsf{DL\text{-}GGM}^{\mathcal{O}}((p, \sigma(1), \sigma(x)), x') = 1 \;\middle|\; x' \leftarrow A^{\mathcal{O}}(p, \sigma(1), \sigma(x))\right] \\ &\leq \Pr\left[P(\mathsf{tr}) = 1 \;\middle|\; x' \xleftarrow{\mathsf{tr}} A^{\mathcal{O}}(p, \sigma(1), \sigma(x))\right] + \frac{1}{p} \ , \end{aligned}$$

where the probability is taken over the random choice of σ and $x \in \mathbb{Z}_p$.

Note that P is a monotonic predicate, therefore, $\mathsf{DL\text{-}GGM}$ is $\frac{1}{p}$-bounded with respect to the predicate P (Definition 9.4). Following the same approach as in Section 9.2, one can show that P is history-oblivious and ϵ-hard where $\epsilon(t) = \frac{1}{2} \cdot \frac{t^2}{p}$. By applying Theorem 9.5, it holds that

$$\Pr\left[x' = x \;\middle|\; x' \leftarrow A^{\mathcal{O}}(p, \sigma(1), \sigma(x))\right] \leq \frac{1}{2} \cdot \frac{\mathbb{E}\left[T_A^2\right]}{p} + \frac{1}{p} \ .$$

□

9.4 Expected-time hardness of SNARKs in the ROM

We show that our framework can be used to derive expected-time hardness of different types of primitives, specifically, the hardness of SNARKs in the ROM. There are several SNARKs in the ROM (with unconditional security) [Mic00, BCS16, CY21a, CY21b]. These constructions follow a shared paradigm: they compile a probabilistic proof (a PCP or an IOP) into a SNARK by using a commitment scheme and other ROM techniques.

Here we focus on the construction of Micali [Mic00] for simplicity. We briefly overview the construction. The Micali construction is based on Kilian's transformation [Kil92] and the Fiat-Shamir paradigm [FS86]. Informally, the argument prover $\mathcal{P}$ commits to a PCP string using a Merkle tree and obtains a root rt. Then, it derives the PCP randomness ρ by querying the random oracle on the root and invokes the PCP verifier $\mathbf{V}_{\mathsf{PCP}}$ with the randomness ρ to obtain the query-answer list $\mathbf{a}$. Finally, it obtains authentication

paths auth for all answers in $\mathbf{a}$ and produces the proof $\pi = (\mathsf{rt}, \mathbf{a}, \mathsf{auth})$. The argument verifier $\mathcal{V}$ parses the proof $\pi = (\mathsf{rt}, \mathbf{a}, \mathsf{auth})$, computes the randomness ρ by querying the random oracle f with rt, checks that the query-answer list $\mathbf{a}$ consists with the authentication paths auth and invokes the PCP verifier $\mathbf{V}_{\mathsf{PCP}}$ with the randomness ρ and the answers $\mathbf{a}$.

Prior work [Mic00, Val08, BCS16] showed that if the underlying PCP has soundness error ϵ_{PCP}, then every malicious prover that makes at most t-queries to the random oracle, can convince the verifier of a false statement with probability at most $\epsilon \leq (t+1) \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{t^2}{2^\lambda}$. Applying our framework, we obtain the following:

Lemma 9.10. *Let PCP be a PCP for a relation $\mathcal{R}$ with soundness error ϵ_{PCP}, and let $\mathsf{Micali}[\mathsf{PCP}, \lambda]$ be the Micali transformation for PCP. For any malicious argument prover $\tilde{\mathcal{P}}$, let $T_{\tilde{\mathcal{P}}}$ be a random variable indicating the number of queries performed by $\tilde{\mathcal{P}}$ until he stops, when given access to a random oracle f that outputs λ bits. Then, for every malicious argument prover $\tilde{\mathcal{P}}$ it holds that*

$$\Pr\left[\begin{array}{l|l} \mathcal{V}^f(\mathbb{x}, \tilde{\pi}) = 1 & f \leftarrow \mathcal{U}(\lambda) \\ \wedge\, \mathbb{x} \notin \mathcal{L}(\mathcal{R}) & (\mathbb{x}, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^f \end{array}\right] \leq (\mathbb{E}\left[T_{\tilde{\mathcal{P}}}\right] + 1) \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{\mathbb{E}\left[T_{\tilde{\mathcal{P}}}^2\right]}{2^\lambda} \ .$$

Proof. We assume that we can derive two distinct oracles from f: an oracle f_{MT} used for the Merkle commitment scheme and an oracle f_ρ used to derive the PCP randomness. We construct the experiment $\mathsf{Micali\text{-}Exp}$ as follows:

$\mathsf{Micali\text{-}Exp}^{f_{\mathsf{MT}}, f_\rho}(\mathbb{x}, \pi)$:

1. Parse π as a tuple $(\mathsf{rt}, \mathbf{a}, \mathsf{auth})$.
2. Derive PCP randomness $\rho := f_\rho(\mathbb{x}, \mathsf{rt})$.
3. Return 1 if and only if $\mathbf{V}_{\mathsf{PCP}}^{[\mathbf{a}]}(\mathbb{x}, \rho) = 1$, $\mathbf{a}$ consists with the authentication paths auth and $\mathbb{x} \notin \mathcal{L}(\mathcal{R})$.

Let $\mathsf{tr}_{\mathsf{MT}} = ((x_1, y_1), \ldots, (x_{t_1}, y_{t_1}))$ be the query-answer list between a malicious argument prover $\tilde{\mathcal{P}}$ and the random oracle f_{MT} and let $\mathsf{tr}_\rho = ((x_1, y_1), \ldots, (x_{t_2}, y_{t_2}))$ be the query-answer list between a malicious argument prover $\tilde{\mathcal{P}}$ and the random oracle f_ρ. Note that each query x to f_{MT} is of the form $x = x^1 \| x^2$. We consider the 3 following predicates:

1. $P_1(\mathsf{tr}_{\mathsf{MT}})$**:** outputs 1 if and only if there exists $i > j$ such that $x_j \neq x_i$ and $y_j = y_i$.
2. $P_2(\mathsf{tr}_{\mathsf{MT}})$**:** outputs 1 if and only if there exists $i > j$ such that $y_i = x_j^1$ or $y_j = x_j^2$.
3. $P_3(\mathsf{tr}_\rho, \mathsf{tr}_{\mathsf{MT}})$**:** outputs 1 if and only if there exists a query $(\mathbb{x}, \mathsf{rt}) \in \mathsf{tr}_\rho$ such that there exists $\mathbf{a}$ where:
   (a) $\mathbf{V}_{\mathsf{PCP}}^{[\mathbf{a}]}(\mathbb{x}, f_\rho(\mathbb{x}, \mathsf{rt})) = 1$; and
   (b) $\mathsf{tr}_{\mathsf{MT}}$ contains valid authentication paths for all answers in $\mathsf{tr}_{\mathsf{MT}}$ with respect to the root rt.

Consider the predicate $P(\mathsf{tr})$, which outputs 1 if and only if one of the above predicates is accepting. We observe that $\mathsf{Micali\text{-}Exp}$ is ϵ_{PCP}-bounded with respect to the predicate P. Given the strict time analysis of the Micali transformation, the predicate P is ϵ-hard where $\epsilon(t) = t \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{t^2}{2^\lambda}$ (for further discussion see [CY21b]).

We argue that P is *history-oblivious*. Note that each query performed by $\tilde{\mathcal{P}}$ is either to f_{MT} or to f_ρ. If $\tilde{\mathcal{P}}$ queried f_{MT}, then P is satisfied only if P_1 or P_2 is accepting. If $\tilde{\mathcal{P}}$ queried f_ρ, then P is satisfied only if P_3 is accepting. The i-th query satisfies the predicate P_1 of P_2 with probability at most $3 \cdot \frac{i-1}{2^\lambda}$ and satisfies the predicate P_3 with probability at most ϵ_{PCP}. Overall, the i-th query satisfies the predicate P with probability at most $\max\{\epsilon_{\mathsf{PCP}}, 3 \cdot \frac{i-1}{2^\lambda}\} \leq \epsilon_{\mathsf{PCP}} + 3 \cdot \frac{i-1}{2^\lambda}$. Thus, we define $\kappa(i) = \epsilon_{\mathsf{PCP}} + 3 \cdot \frac{i-1}{2^\lambda}$, and get that for any trace of length i (which for the predicate outputs 0), and any query x it holds that

$$\Pr\left[P\left(\mathsf{tr} \| (x, y)\right) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ y = f(x) \end{array}\right] \leq \kappa(i) \ .$$

Then, we have that

$$\sum_{j=0}^{t} \kappa(j) = \sum_{j=0}^{t} \epsilon_{\mathsf{PCP}} + 3 \cdot \frac{i-1}{2^{\lambda}} \leq t \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{t^2}{2^{\lambda}} = \epsilon(t)$$

Overall, we conclude that the predicate is history-oblivious.

By applying Theorem 9.5 we derive the following

$$\begin{aligned}
&\Pr\left[\mathcal{V}^{f}(\mathbb{x}, \tilde{\pi}) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{x}, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^{f} \end{array}\right] \\
&= \Pr\left[\mathsf{Micali\text{-}Exp}^{f}(\mathbb{x}, \tilde{\pi}) = 1 \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{x}, \tilde{\pi}) \leftarrow \tilde{\mathcal{P}}^{f} \end{array}\right] \\
&\leq \Pr\left[P(\mathsf{tr}) \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ (\mathbb{x}, \tilde{\pi}) \overset{\mathsf{tr}}{\leftarrow} \tilde{\mathcal{P}}^{f} \end{array}\right] + \epsilon_{\mathsf{PCP}} \\
&\leq \mathbb{E}\left[T_{\tilde{\mathcal{P}}}\right] \cdot \epsilon_{\mathsf{PCP}} + \frac{3}{2} \cdot \frac{\mathbb{E}\left[T_{\tilde{\mathcal{P}}}^{2}\right]}{2^{\lambda}} + \epsilon_{\mathsf{PCP}} \;\;,
\end{aligned}$$

which completes the proof. □

References

[AAB+02] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology – EUROCRYPT '02*, pages 418–433, 2002.

[ACK21] T. Attema, R. Cramer, and L. Kohl. A compressed Σ-protocol theory for lattices. In T. Malkin and C. Peikert, editors, *Advances in Cryptology – CRYPTO '21*, volume 12826 of *Lecture Notes in Computer Science*, pages 549–579. Springer, 2021.

[AHK20] T. Agrikola, D. Hofheinz, and J. Kastner. On instantiating the algebraic group model from falsifiable assumptions. In *Advances in Cryptology – EUROCRYPT '20*, pages 96–126, 2020.

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology – EUROCRYPT '16*, pages 327–357, 2016.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, pages 31–60, 2016.

[BD20] M. Bellare and W. Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In *Progress in Cryptology – INDOCRYPT '20*, pages 529–552, 2020.

[BD21] M. Bellare and W. Dai. Chain reductions for multi-signatures and the HBMS scheme. In *Advances in Cryptology – ASIACRYPT '21*, pages 650–678, 2021.

[BDN18] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology – ASIACRYPT '18*, pages 435–464, 2018.

[BFL20] B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In *Advances in Cryptology – CRYPTO '20*, pages 121–151, 2020.

[BN06] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 390–399, 2006.

[CD98] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO '98*, pages 424–441, 1998.

[CFG+15] D. Catalano, D. Fiore, R. Gennaro, and K. Vamvourellis. Algebraic (trapdoor) one-way functions: Constructions and applications. *Theoretical Computer Science*, 592:143–165, 2015.

[Cra96] R. Cramer. Modular design of secure yet practical cryptographic protocols. *PhD thesis, CWI and University of Amsterdam*, 1996.

[CY21a] A. Chiesa and E. Yogev. Subquadratic SNARGs in the random oracle model. In *Proceedings of the 41st Annual International Cryptology Conference*, CRYPTO '21, pages 711–741, 2021.

[CY21b] A. Chiesa and E. Yogev. Tight security bounds for Micali's SNARGs. In *Proceedings of the 19th Theory of Cryptography Conference*, TCC '21, pages 401–434, 2021.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology – CRYPTO '18*, pages 33–62, 2018.

[FPS20] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In *Advances in Cryptology – EUROCRYPT '20*, pages 63–95, 2020.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86*, pages 186–194, 1986.

[GLS+04] R. Gennaro, D. Leigh, R. Sundaram, and W. S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *Advances in Cryptology – ASIACRYPT '04*, pages 276–292, 2004.

[GQ88] L. C. Guillou and J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 1988.

[HL10] C. Hazay and Y. Lindell. Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography. Springer, 2010.

[JT20] J. Jaeger and S. Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 414–443, 2020.

[Kil92] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732, 1992.

[KL08] J. Katz and Y. Lindell. Handling expected polynomial-time strategies in simulation-based security proofs. *Journal of Cryptology*, 21(3):303–349, 2008.

[KMP16] E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *Advances in Cryptology – CRYPTO '16*, pages 33–61, 2016.

[Mau05] U. Maurer. Abstract models of computation in cryptography. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, pages 1–12, 2005.

[Mic00] S. Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MPS+19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[MTT19] T. Mizuide, A. Takayasu, and T. Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In *Topics in Cryptology – CT-RSA '19*, pages 169–188, 2019.

[NRS21] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In *Advances in Cryptology – CRYPTO '21*, pages 189–221, 2021.

[Oka92] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology – CRYPTO '92*, pages 31–53, 1992.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

[RS20] L. Rotem and G. Segev. Algebraic distinguishers: From discrete logarithms to decisional Uber assumptions. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 366–389, 2020.

[RS21] L. Rotem and G. Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In T. Malkin and C. Peikert, editors, *Advances in Cryptology – CRYPTO '21*, volume 12825 of *Lecture Notes in Computer Science*, pages 222–250. Springer, 2021.

[Sch89] C. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – CRYPTO '89*, pages 239–252, 1989.

[Sch91] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, 1997.

[Val08] P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In R. Canetti, editor, *Proceedings of the 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[Yun15] A. Yun. Generic hardness of the multiple discrete logarithm problem. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT '15*, volume 9057 of *Lecture Notes in Computer Science*, pages 817–836. Springer, 2015.