Expected-Time Cryptography: Generic Techniques and Applications to Concrete Soundness

Joseph Jaeger(✉) and Stefano Tessaro

Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, USA
{jsjaeger,tessaro}@cs.washington.edu

Abstract. This paper studies concrete security with respect to *expected-time* adversaries. Our first contribution is a set of generic tools to obtain tight bounds on the advantage of an adversary with expected-time guarantees. We apply these tools to derive bounds in the random-oracle and generic-group models, which we show to be tight.

As our second contribution, we use these results to derive concrete bounds on the soundness of public-coin proofs and arguments of knowledge. Under the lens of concrete security, we revisit a paradigm by Bootle et al. (EUROCRYPT '16) that proposes a general Forking Lemma for multi-round protocols which implements a rewinding strategy with expected-time guarantees. We give a tighter analysis, as well as a modular statement. We adopt this to obtain the first quantitative bounds on the soundness of Bulletproofs (Bünz et al., S&P 2018), which we instantiate with our expected-time generic-group analysis to surface inherent dependence between the concrete security and the statement to be proved.

Keywords: Concrete security · Proof systems

1 Introduction

Cryptography usually adopts a *worst-case* angle on complexity. For example, in the context of *concrete* security, a typical theorem shows that an adversary running for at most t steps succeeds with advantage at most ε. In this paper, we instead study the concrete security of cryptographic schemes and assumptions as a function of the *expected* running time of the adversary.

Expected-time complexity is a natural measure in its own right – e.g., it is very common in cryptanalysis, as it is often much easier to analyze. But it is also a useful technical tool – indeed, simulators and extractors are often expected time, sometimes inherently so [1]. To use these technical tools, we need assumptions to hold with respect to expected time.

The problem has been studied closely by Katz and Lindell [14], who also suggest expected-time adversaries as a natural model, which however also comes with several technical challenges. Either way, the resulting common wisdom is

R. Pass and K. Pietrzak (Eds.): TCC 2020, LNCS 12552, pp. 414–443, 2020.
https://doi.org/10.1007/978-3-030-64381-2_15

that assumptions which are true with respect to (non-uniform) worst-case polynomial time are true for expected polynomial-time, and often more fine-grained statements are possible via Markov's inequality (see below). However, for concrete security, such generic argument fail to give tight bounds.

Summary of contributions. This paper makes progress on two fronts.

First, as our main technical contribution, we introduce general tools to give *tight* concrete security bounds in information-theoretic settings (e.g., in the random-oracle or generic-group models) for expected-time adversaries. Our tools can easily translate many existing proofs from the worst-case to the expected-time regime. We derive for example tight bounds for finding collisions in a random oracle, for the PRF security of random oracles, and for computing discrete logarithms in the generic-group model. We also obtain bounds for the security of key-alternating ciphers against expected-time adversaries.

Second, we study a "Forking Lemma" to prove soundness of *multi-round* public-coin proofs and arguments (of knowledge) satisfying a generalized notion of *special soundness*, enabling witness extraction from a suitable tree of accepting interactions. In particular, we follow a blueprint by Bootle *et al.* [6], which has also been adopted by follow-up works [7,8,24]. In contrast to prior works, we provide a concrete analysis of the resulting expected-time witness extraction strategy, and also give a modular treatment of the techniques which may be of independent interest.

We showcase these tools by deriving concrete bounds for the soundness of Bulletproofs [7] in terms of the expected-time hardness of solving the discrete logarithm problem. Instantiating the bound with our generic-group model analysis will in particular illustrate the dependence of soundness on group parameters and on the complexity of the statement to be proved. We are unaware of any such result having been proved, despite the practical appeal of these protocols.

The remainder of this introduction provides a detailed overview of our results.

1.1 Information-Theoretic Bounds for Expected-Time Adversaries

Our first contribution is a framework to prove tight bounds with respect to expected-time adversaries. We focus on *information-theoretic analyses*, such as those in the random oracle [3] and the generic group [18,22] models.

Our focus on tight bounds is what makes the problem hard. Indeed, one can usually obtain a non-tight bound using Markov's inequality. For example, the probability $\varepsilon(T, N)$ of a T-time adversary finding a collision in a random oracle with N outputs satisfies $\varepsilon(T, N) \leqslant T^2/2N$, and this bound is tight. If we instead aim to upper bound the probability $\varepsilon(\mu_T, N)$ of finding a collision for an adversary that runs in *expected* time $\mu_T = \mathsf{E}[T]$, Markov's inequality yields, for every $T^* > \mu_T$,

$$\varepsilon(\mu_T, N) \leqslant \mathsf{Pr}\left[T > T^*\right] + \frac{(T^*)^2}{2N} \leqslant \frac{\mu_T}{T^*} + \frac{(T^*)^2}{2N} \leqslant 2 \cdot \sqrt[3]{\frac{\mu_T^2}{2N}}\,, \tag{1}$$

where the right-most inequality is the result of setting T^* such that $\frac{\mu_T}{T^*} = \frac{(T^*)^2}{2N}$. Here, we prove the better upper bound

$$\varepsilon(\mu_T, N) \leqslant \sqrt{\frac{\mu_T^2}{2N}}\,, \tag{2}$$

as a corollary of the techniques we introduce below. This bound is tight: To see this, take an adversary which initially flips a biased coin, which is heads with probability $\mu_T/\sqrt{N}$. If the coin is tails, it aborts, failing to find a collision. If the coin is heads, it makes $\sqrt{N}$ queries to find a collision with high probability. Then, this adversary succeeds with probability $\Omega(\mu_T/\sqrt{N}) = \Omega(\sqrt{\mu_T^2/N})$, and its expected run time is μ_T.

Both (1) and (2) show that $\mu_T \geqslant \Omega(\sqrt{N})$ must hold to find a collision *with probability one*. However, exact probability bounds are important in the regime $\mu_T = o(\sqrt{N})$. For example, say we are asked to find a collision in at least one out of u independent random oracles, and the expected number of queries to each is μ_T. Then, a hybrid argument bounds the probability by $u \cdot \varepsilon(\mu_T, N)$, making the difference between a square-root and a cube-root bound on $\varepsilon(\mu_T, N)$ important.

A Generic Approach for bad-flag analyses. We aim for a general approach to transform information-theoretic bounds for worst-case query complexity into bounds with respect to *expected* query complexity. If an existing analysis (with respect to worst-case complexity) follows a certain pattern, then we easily obtain an expected query complexity bound.

More concretely, many security proofs follow the "equivalent-until-bad" format (as formalized by Bellare and Rogaway [4], but equivalent formulations can be derived from the works of Maurer [17] and Shoup [23]). The goal here is to upper bound the advantage of an adversary $\mathcal{A}$ distinguishing two games G_0 and G_1, which behave *identically* until some bad flag bad is set. Then, the distinguishing advantage is upper bounded by the probability of setting bad to true – an event we denote as $\mathsf{BAD}^{\mathcal{A}}$. Typically, G_0 is the "real world" and G_1 is the "ideal world". Now, let Q_1 be the number of queries by an adversary $\mathcal{A}$ in G_1, which is a random variable. Then, we say that this game pair satisfies δ-*boundedness* if

$$\mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{A}} \mid Q_1 = q\right] \leqslant \delta(q)$$

for all $q \geqslant 1$ and adversaries $\mathcal{A}$. This condition is not without loss of generality, but it can be ensured in all examples we verified.

Our first main theorem (Theorem 1) shows that if $\delta(q) = \Delta \cdot q^d/N$, then the probability of setting $\mathsf{BAD}^{\mathcal{A}}$ (in either of the two games), and hence the advantage of distinguishing G_0 and G_1, is upper bounded as

$$\mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{A}}\right] \leqslant 5 \cdot \left(\frac{\Delta \mathsf{E}[Q_0]^d}{N}\right)^{1/d},$$

where (quite) crucially Q_0 is the number of queries of $\mathcal{A}$ in G_0. This asymmetry matters in applications - we typically measure complexity in the *real world*, but

δ-boundedness only holds in the ideal world.

Proof idea. The key step behind the proof of Theorem 1 is the introduction of an *early-terminating* adversary $\mathcal{B}$, which behaves as $\mathcal{A}$ in attempting to set bad, but aborts early after $U = \left\lfloor \sqrt[d]{Nu/\Delta} \right\rfloor = \Theta(\sqrt[d]{N/\Delta})$ queries, where $u = 2^{-d}$. One can then show that (we can think of the following probabilities in G_0)

$$\mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{A}}\right] \leqslant \mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{B}}\right] + \mathsf{Pr}\left[Q_0 > U\right] ,$$

because $\mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{A}} \wedge Q_0 \leqslant U\right] \leqslant \mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{B}}\right]$. Markov's inequality then yields

$$\mathsf{Pr}\left[Q_0 > U\right] \leqslant \frac{\mathsf{E}\left[Q_0\right]}{U} = \Theta\left(\sqrt[d]{\Delta \mathsf{E}\left[Q_0\right]^d / N}\right) ,$$

which is of the right order.

Therefore, the core of the proof is to show $\mathsf{Pr}\left[\mathsf{BAD}^{\mathcal{B}}\right] = O\left(\sqrt[d]{\Delta \mathsf{E}\left[Q_0\right]^d / N}\right)$. This will require using δ-boundedness first, but a careful reader may observe that this will only upper bound the probability with respect to $\mathsf{E}\left[Q_1\right]$, and not $\mathsf{E}\left[Q_0\right]$. The bulk of the proof is then to switch between the two.

Examples. We apply the above framework to a few examples, to show its applicability. We show bounds on the hardness of discrete logarithms in the generic-group model [18,22], and on the collision-resistance and PRF security of random oracles. In particular, our framework also works for notions which are not indistinguishability based, such as collision-resistance of a random oracle, by introducing a suitable world G_1 where it is hard to win the game.

The H-Coefficient method. Equivalent-until-bad analyses are not always the simplest way to prove security (despite the fact that in principle every analysis can be cast in this format, as shown in [19]). We also give a variant of the above approach tailored at proving security in a simpler version of the H-coefficient method [9,20] which considers what is referred to as *pointwise-proximity* in [12]. This amounts to using the standard H-coefficient method without bad transcripts. (To the best of our knowledge, this simpler version of the method is due to Bernstein [5].) This allows us to obtain expect-time versions of security bounds for the PRF/PRP switching lemma and for key-alternating ciphers, the latter building on top of work by Hoang and Tessaro [12].We defer details on this to the full version of this paper [13].

1.2 Forking Lemmas and Concrete Soundness

One motivation for studying expected-time adversaries is as a *tool* to prove bounds for worst-case complexity, rather than as a goal on itself. We expose here one such application in the context of proving *soundness* bounds for public-coin

proofs/arguments (of knowledge). In particular, soundness/proof-of-knowledge proofs for several protocols (like [6–8,24]) rely on generalizations of the Forking Lemma (originally proposed by Pointcheval and Stern [21] for three-round protocols) which adopt expected-time witness extraction strategies. These have only been analyzed in an asymptotic sense, and our goal is to give a concrete-security treatment. We propose here a modular treatment of these techniques, and instantiate our framework to provide concrete bounds on the soundness of Bulletproofs [7], a succinct proof system which has enjoyed wide popularity.

Forking Lemmas. Pointcheval and Stern's original "Forking Lemma" [21] deals with Σ-protocols that satisfy *special soundness* - these are three-round protocols, where a transcript takes the form (a, c, d), with c being the verifier's single *random* challenge. Here, given common input x, the prover P proves knowledge to V of a witness w for a relation R. The proof of knowledge property is proved by giving an *extractor* $\mathcal{B}$ which produces a witness for x given (black-box) access to a prover P^* – if P^* succeeds with probability ε, then $\mathcal{B}$ succeeds with probability (roughly) ε^2. Concretely, $\mathcal{B}$ simulates an execution of P^* with a random challenge c, which results in a transcript (a, c, d), and then rewinds P^* to just before obtaining c, and feeds a different challenge c' to obtain a transcript (a, c', d'). If both transcripts are accepting, and $c \neq c'$, a witness can be extracted via special soundness. Bellare and Neven [2] give alternative Forking Lemmas where $\mathcal{B}$'s success probability approaches ε, at the cost of a larger running time.

Expected-time extraction. It is natural to expect that the success probability of $\mathcal{B}$ above degrades exponentially in the number of required accepting transcripts. Crucially, however, one can make the Forking Lemma tight with respect to probability if we relax $\mathcal{B}$ to have bounded *expected* running time. Now, $\mathcal{B}$ runs P^* once with a random challenge c and, if it generates a valid transcript (a, c, d), we rewind P^* to before receiving the challenge c, and keep re-running it from there with fresh challenges until we obtain a second valid transcript (a, c', d') for $c \neq c'$. The expected running time is only *twice* that of P^*.

A general Forking Lemma. An extension of this idea underlies the analysis of recent succinct public-coin multi-round interactive arguments of knowledge [6–8,24], following a workflow introduced first by Bootle et al. (BCCGP) [6] which extracts a witness from a *tree* of multi-round executions obtained by clever rewinding of P^*. In particular, since the number of generated accepted interactions is large (i.e., exponential in the number of rounds), the usage of an expected-time strategy is essential to extract with good enough probability.

These works in fact prove the stronger property of *witness-extended emulation* [11,16]. This means that with black-box access to a prover P^*, an expected-time *emulator* E (1) generates a transcript with the same distribution as in an interaction between P^* and the verifier V, *and* (2) if this transcript is accepting, then a valid witness is produced along with it. In the case of *arguments*, it is possible that (2) fails, but this would imply breaking an underlying assumption.

The BCCGP framework was refined in follow-up works [7,8,24], but these remain largely asymptotic. We give here a clean and modular treatment of the

BCCGP blueprint, which makes it amenable to a concrete security treatment. This will in particular require using our tools from the first part of the paper to analyze the probability that we generate a well-formed tree of transcripts from which a witness can be generated. We believe this to be of independent interest.

In the full version of this paper [13], we compare this expected-time forking lemma to one with strict running-time guarantees and confirm that the expected-time approach achieves a clear benefit in terms of tightness of the reduction.

Application to Bulletproofs. Finally, we apply the above framework to obtain a bound on the *concrete soundness* for public-coin interactive argument systems, and focus on Bulletproofs [7][1]. We obtain a bound in terms of the expected-time hardness of the discrete logarithm problem, and we combine this with our generic-group analysis to get a bound on the soundness in the generic-group model[2]. Of independent interest, the result relies on a *tight* reduction of finding non-trivial discrete log relations to the plain discrete log problem – which we give in Lemma 3.

Our bound is in particular on the probability $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*)$ of a cheating prover P^* convincing a verifier V (from proof system PS) on input x generated by a (randomized) instance *generator* G, and we show that

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*) \leqslant \mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B}) + O\left(\frac{q_{\mathsf{P}^*} \cdot LM^3 \log_2(M)}{\sqrt{|\mathbb{G}|}}\right) ,$$

where q_{P^*} measures the number of group operations by P^*, M is the number of multiplication gates for a circuit representing the relation R, L is a parameter of that circuit (which we assume is small for this discussion, but may be as large as $2M$), $\mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B})$ is the probability of $\mathcal{B}$ extracting a witness w for an x sampled by G, where $\mathcal{B}$ is an extractor whose (expected) running time amounts to roughly M^3 that of P^*.

This bound is interesting because it highlights the dependence of the soundness probability on the group size $|\mathbb{G}|$ and on M. It in fact shows that for typical instantiations, where $|\mathbb{G}| \approx 2^{256}$, the guaranteed security level is fairly low for modest-sized circuits (say with $M = 2^{20}$). It is a good question whether this bound can be made tighter, in particular with respect to its dependence on M.

We also note that for specific instance generators G our tools may be helpful to estimate $\mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B})$.

[1] Our focus is somewhat arbitrary, and motivated by the popularity of this proof system.

[2] This bound is helped by the fact that our casting of the generic-group model allows multi-exponentiations $(g_0, \ldots, g_n, a_0, \ldots, a_n \to \prod_{i=0}^{n} g_i^{a_i})$ as a unit operation. This does not change the derived bound in the generic-group model, while decreasing the number of generic-group queries made by the Bulletproofs verifier.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{N}_{>0} = \mathbb{N} \setminus \{0\}$. For $N \in \mathbb{N}$, let $[N] = \{1, 2, \dots, N\}$. For $j > k$ we adopt the conventions that $\prod_{i=j}^{k} n_i = 1$ and $(m_j, m_{j+1}, \dots, m_k) = ()$. Equivalence mod p is denoted $\equiv_p$.

We let $\mathsf{Perm}(S)$ denote the set of all permutations on set S and $\mathsf{Fcs}(S, S')$ denote the set of all functions from S to S'. Sampling x uniformly from the set S is denoted $x \leftarrow\!\!\$\, S$. The notation $S = S' \sqcup S''$ means that $S = S' \cup S''$ and $S' \cap S'' = \emptyset$, i.e., S' and S'' partition S. We let $\{0,1\}^*$ denote the set of finite-length bitstrings and $\{0,1\}^\infty$ denote the set of infinite-length bitstrings.

We let $y \leftarrow \mathcal{A}^{\mathrm{O}}(x_1, x_2, \dots; c)$ denote the execution of $\mathcal{A}$ on input $x_1, x_2, \dots$ and coins $c \in \{0,1\}^\infty$ with access to oracle(s) O, producing output y. When c is chosen uniformly we write $y \leftarrow\!\!\$\, \mathcal{A}^{\mathrm{O}}(x_1, x_2, \dots)$. For a stateful algorithm $\mathcal{A}$ with state s we use $y \leftarrow \mathcal{A}^{\mathrm{O}}(x_1, x_2, \cdots : s; c)$ as shorthand for the expression $(y, s) \leftarrow A^{\mathrm{O}}(x_1, x_2, \dots, s; c)$. When some of an algorithm's output is not going to be used we will write $\cdot$ in place of giving it a variable name.

We use pseudocode games, inspired by the code-based game framework of Bellare and Rogaway [4]. See Fig. 1 for some example games. If H is a game, then $\mathsf{Pr}[\mathsf{H}]$ denotes the probability that it outputs true. We use $\wedge$, $\vee$, $\Leftrightarrow$, and $\neg$ for the logical operators "and", "or", "iff", and "not".

Running-time conventions. The most commonly used notion for the running time of an algorithm is worst-case. For this, one first fixes a computational model with an associated notion of computational steps. Then an algorithm $\mathcal{A}$ has worst-case running time t if for all choice of $x_1, x_2, \dots$ and c it performs at most t computation steps in the execution $\mathcal{A}^{\mathrm{O}}(x_1, x_2, \dots; c)$, no matter how O responds to any oracle queries $\mathcal{A}$ makes.

In this paper we are interested in proving bounds that instead depend on the expected number of computation steps that $\mathcal{A}$ performs. There may be randomness in how the inputs $x_1, x_2, \dots$ to $\mathcal{A}$ and the responses to O queries are chosen (in addition to the random selection of c).

There is more variation in how expected running time may be defined. We will provide our bounds in terms of the expected running time of adversaries interacting with the "real" world that they expect to interact with. Such a notion of expected runtime is brittle because the expected runtime of the adversary may vary greatly when executing in some other world; however, this notion is the strongest for the purposes of our results because it will guarantee the same bounds for notions of expected running time which restrict the allowed adversaries more. See [10,15] for interesting discussion of various ways to define expected polynomial time.

For many of the results of this paper, rather than directly measuring the runtime of the adversary we will look at the (worst-case or expected) number of oracle queries that it makes. The number of oracle queries can, of course, be upper bounded by the number of computational steps.

Game $\mathsf{H}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A})$	Game $\mathsf{H}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A})$	Adversary $\mathcal{C}(g,h)$
$g \leftarrow\!\!\$\ \mathbb{G}^*$	$\boldsymbol{g} \leftarrow (g_0,\ldots,g_n) \leftarrow\!\!\$\ \mathbb{G}^n$	For $i = 0,\ldots,n$
$h \leftarrow\!\!\$\ \mathbb{G}$	$(a_0,\ldots,a_n) \leftarrow\!\!\$\ \mathcal{A}(\mathbf{g})$	$\quad x_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $y_i \leftarrow\!\!\$\ \mathbb{Z}_p$
$a \leftarrow\!\!\$\ \mathcal{A}(g,h)$	If $\forall i, a_i \equiv_p 0$ then	$\quad g_i \leftarrow g^{x_i} \cdot h^{y_i}$
Return $(g^a = h)$	$\quad$ Return false	$(a_0,\ldots,a_n) \leftarrow\!\!\$\ \mathcal{A}((g_0,\ldots,g_n))$
	Return $(\prod_{i=0}^{n} g_i^{a_i} = 1_{\mathbb{G}})$	If $\sum_i a_i y_i \equiv_p 0$ then return 0
		Else return $-\sum_i a_i x_i / \sum_i a_i y_i$

Fig. 1. Left: Game defining discrete log security of group $\mathbb{G}$. **Middle:** Game defining discrete log relation security of group $\mathbb{G}$. **Right:** Reduction adversary for Lemma 3.

Useful lemmas. We will make use of Markov's inequality and the Schwartz-Zippel Lemma, which we reproduce here.

Lemma 1 (Markov's Inequality). *Let X be a non-negative random variable and $c > 0$ be a non-negative constant, then*

$$\mathsf{Pr}[X > c] \leqslant \mathsf{Pr}[X \geqslant c] \leqslant \mathsf{E}[X]/c.$$

Lemma 2 (Schwartz-Zippel Lemma). *Let $\mathbb{F}$ be a finite field and let $p \in \mathbb{F}[x_1, x_2, \ldots x_n]$ be a non-zero polynomial with degree $d \geqslant 0$. Then*

$$\mathsf{Pr}[p(r_1,\ldots,r_n) = 0] \leqslant d/|\mathbb{F}|$$

where the probability is over the choice of $r_1,\ldots,r_n$ according to $r_i \leftarrow\!\!\$\ \mathbb{F}$.

Discrete Logarithm Assumptions. Let $\mathbb{G}$ be a cyclic group of prime order p with identity $1_{\mathbb{G}}$ and $\mathbb{G}^* = \mathbb{G}\backslash\{1_{\mathbb{G}}\}$ be its set of generators. Let $(g_0,\ldots,g_n) \in \mathbb{G}^n$ and $(a_0,\ldots,a_n) \in \mathbb{Z}_p$. If $\prod_{i=0}^{n} g_i^{a_i} = 1_{\mathbb{G}}$ and a least one of the a_i are non-zero, this is said to be a non-trivial discrete log relation. It is believed to be hard to find non-trivial discrete log relations in cryptographic groups (when the g_i are chosen at random). We refer to computing $\prod_{i=0}^{n} g_i^{a_i}$ as a multi-exponentiation of size $n+1$.

Discrete log relation security is defined by the game in the middle of Fig. 1. In it, the adversary $\mathcal{A}$ is given a vector $\boldsymbol{g} = (g_0,\ldots,g_n)$ and attempts to find a non-trivial discrete log relation. We define $\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A})]$. Normal discrete log security is defined by the game in the left panel of Fig. 1. In it, the adversary attempts to find the discrete log of $h \in \mathbb{G}$ with respect to a generator $g \in \mathbb{G}^*$. We define $\mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{A})]$.

It is well known that discrete log relation security is asymptotically equivalent to discrete log security. The following lemma makes careful use of self-reducibility techniques to give a concrete bound showing that discrete log relation security is tightly implied by discrete log security.

Lemma 3. *Let $\mathbb{G}$ be a group of prime order p and $n \geqslant 1$ be an integer. For any $\mathcal{B}$, define $\mathcal{C}$ as shown in Fig. 1. Then*

$$\mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{B}) \leqslant \mathsf{Adv}^{\mathsf{dl}}_{\mathbb{G}}(\mathcal{C}) + 1/p.$$

Game $\mathsf{G}_b^{(G,G')}(\mathcal{A})$	$\textsc{Orac}(x)$
$c_0 \leftarrow\!\!\$\ \{0,1\}^\infty$	$t \leftarrow t+1$
$c_1 \leftarrow\!\!\$\ \{0,1\}^\infty$	If $\neg\mathsf{bad}$ then
$c_{\mathcal{A}} \leftarrow\!\!\$\ \{0,1\}^\infty$	$\quad \mathsf{bad}_t \leftarrow G'(x : s'; c_0, c_1)$
$t \leftarrow 0$	$\quad$ If bad_t then $\mathsf{bad} \leftarrow \mathsf{true}$
$\mathsf{bad} \leftarrow \mathsf{false}$	If bad then $d \leftarrow b$
$s \leftarrow \varepsilon$	Else $d \leftarrow 1$
$s' \leftarrow \varepsilon$	$y \leftarrow G(d, x : s; c_1, c_d)$
Run $\mathcal{A}^{\textsc{Orac}}(c_{\mathcal{A}})$	Return y

Fig. 2. Identical-until-bad games defined from game specification (G, G').

The runtime of $\mathcal{C}$ is that of $\mathcal{B}$ plus the time to perform $n+1$ multi-exponentiations of size 2 and some computations in the field $\mathbb{Z}_p$.

The proof of this theorem is deferred to the full version of the paper [13].

3 Bad Flag Analysis for Expected-Time Adversaries

In this section we show how to (somewhat) generically extend the standard techniques for analysis of "bad" flags from worst-case adversaries to expected-time adversaries. Such analysis is a fundamental tool for cryptographic proofs and has been formalized in various works [4,17,23]. Our results are tailored for the setting where the analysis of the bad flag is information theoretic (e.g., applications in ideal models), rather than reliant on computational assumptions.

We start by introducing our notation and model for identical-until-bad games in Sect. 3.1. Then in Sect. 3.2 we give the main theorem of this section which shows how to obtain bounds on the probability that an expected time adversary causes a bad flag to be set. Finally, in Sect. 3.3 we walk through some basic applications (collision-resistance and PRF security in the random oracle model and discrete log security in the generic group model) to show the analysis required for expected time adversaries follows from simple modifications of the techniques used for worst-case adversaries.

3.1 Notation and Experiments for Identical-Until-Bad Games

Identical-until-bad games. Consider Fig. 2 which defines a pair of games $\mathsf{G}_0^{(G,G')}$ and $\mathsf{G}_1^{(G,G')}$ from a *game specification* (G, G'). Here G and G' are stateful randomized algorithms. At the beginning of the game, coins c_0, c_1, and $c_{\mathcal{A}}$ are sampled uniformly at random[3]. The first two of these are used by G and G'

[3] In the measure-theoretic probability sense with each individual bit of the coins being sampled uniformly and independently.

while the last is used by $\mathcal{A}$[4]. The counter t is initialized to 0, the flag bad is set to false, and states s and s' are initialized for use by G and G'.

During the execution of the game, the adversary $\mathcal{A}$ repeatedly makes queries to the oracle ORAC. The variable t counts how many queries $\mathcal{A}$ makes. As long as bad is still false (so $\neg$bad is true), for each query made by $\mathcal{A}$ the algorithm G' will be given this query to determine if bad should be set to true. When $b = 1$, the behavior of ORAC does not depend on whether bad is set because the output of the oracle is always determined by running $G(1, x : s; c_1, c_1)$. When $b = 0$, the output of the oracle is defined in the same way up until the point that bad is set to true. Once that occurs, the output is instead determined by running $G(0, x : s; c_1, c_0)$. Because these two games are identical except in the behavior of the code $d \leftarrow b$ which is only executed once bad = true, they are "identical-until-bad".

In this section, the goal of the adversary is to cause bad to be set to true. Bounding the probability that $\mathcal{A}$ succeeds in this can be used to analyze security notions in two different ways. For indistinguishability-based security notions (e.g., PRG or PRF security), the two games G_b would correspond to the two worlds the adversary is attempting to distinguish between. For other security notions (e.g., collision resistance or discrete log security), we think of one of the G_b as corresponding to the game the adversary is trying to win and the other as corresponding to a related "ideal" world in which the adversary's success probably can easily be bounded. In either case, the fundamental lemma of game playing [4] can be used to bound the advantage of the adversary using a bound on the probability that bad is set.

A COMBINED EXPERIMENT. For our coming analysis it will be useful to relate executions of $\mathsf{G}_0^{(G,G')}(\mathcal{A})$ and $\mathsf{G}_1^{(G,G')}(\mathcal{A})$ to each other. For this we can think of a single combined experiment in which we sample c_0, c_1, and $c_{\mathcal{A}}$ *once* and then run both games separately using these coins.

For $b \in \{0, 1\}$, we let $Q_b^{\mathcal{A}}$ be a random variable denoting how many oracle queries $\mathcal{A}$ makes in the execution of $\mathsf{G}_b^{(G,G')}(\mathcal{A})$ during this experiment. We let $\mathsf{BAD}_t^{\mathcal{A}}[b]$ denote the event that G' sets bad_t to true in the execution of $\mathsf{G}_b^{(G,G')}(\mathcal{A})$. Note that $\mathsf{BAD}_t^{\mathcal{A}}[0]$ will occur if and only if $\mathsf{BAD}_t^{\mathcal{A}}[1]$ occurs, because the behavior of both games are identical up until the first time that bad is set and G' is never again executed once bad is true. Hence we can simplify notation by defining $\mathsf{BAD}_t^{\mathcal{A}}$ to be identical to the event $\mathsf{BAD}_t^{\mathcal{A}}[0]$, while keeping in mind that we can equivalently think of this event as occurring in the execution of either game. We additionally define the event that bad is ever set $\mathsf{BAD}^{\mathcal{A}} = \bigvee_{i=1}^{\infty} \mathsf{BAD}_i^{\mathcal{A}}$, the event that bad is set by one of the first j queries the adversary makes $\mathsf{BAD}_{\leqslant j}^{\mathcal{A}} = \bigvee_{i=1}^{j} \mathsf{BAD}_j^{\mathcal{A}}$, and the event that bad is set after the j-th query the adversary makes $\mathsf{BAD}_{>j}^{\mathcal{A}} = \bigvee_{i=j+1}^{\infty}$. Clearly, $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] = \mathsf{Pr}[\mathsf{BAD}_{\leqslant j}^{\mathcal{A}}] + \mathsf{Pr}[\mathsf{BAD}_{>j}^{\mathcal{A}}]$. Again we can equivalently think of these events as occurring in either game. When the

[4] We emphasize that these algorithms are not allowed any randomness beyond the use of these coins.

adversary is clear from context we may choose to omit it from the superscript in our notation.

The fact that both games behave identically until bad is set true allows us to make several nice observations. If BAD does not hold, then $Q_0 = Q_1$ must hold. If BAD_t holds for some t, then both Q_0 and Q_1 must be at least t. One implication of this is that if $Q_1 = q$ holds for some q, then BAD is equivalent to $\mathsf{BAD}_{\leqslant q}$. Additionally, we can see that $\mathsf{Pr}[\mathsf{BAD}_{>q}] \leqslant \mathsf{Pr}[Q_b > q]$ must hold.

Defining our events and random variables in this single experiment will later allow to consider the expectation $\mathsf{E}[Q_0^d | Q_1 = q]$ for some $d, q \in \mathbb{N}$. In words, that is the expected value of Q_0 raised to the d-th power conditioned on $c_0, c_1, c_{\mathcal{A}}$ having been chosen so that $Q_1 = q$ held. Since Q_0 and Q_1 can only differ if BAD occurs we will be able to use $\mathsf{Pr}[\mathsf{BAD} | Q_1 = q]$ to bound how far $\mathsf{E}[Q_0^d | Q_1 = q]$ can be from $\mathsf{E}[Q_1^d | Q_1 = q] = q^d$.

δ-BOUNDEDNESS. Existing analysis of identical-until-bad games is done by assuming a worst-case bound $q_{\mathcal{A}}$ on the number of oracle queries that $\mathcal{A}$ makes (in either game). Given such a bound, one shows that $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] \leqslant \delta(q_{\mathcal{A}})$ for some function δ. We will say that a game specification (G, G') is δ-bounded if for all $\mathcal{A}$ and $q \in \mathbb{N}$ we have that

$$\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}} | Q_1 = q] \leqslant \delta(q).$$

As observed earlier, if $Q_1 = q$ holds then bad_t cannot be set for any $t > q$. Hence $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}} | Q_1 = q] = \mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}_{\leqslant q} | Q_1 = q]$.

We will, in particular, be interested in that case that $\delta(q) = \Delta \cdot q^d / N$ for some $\Delta, d, N \geqslant 1$[5]. We think of Δ and d as "small" and of N as "large". The main result of this section bounds the probability that an adversary sets bad by $O\left(\sqrt[d]{\delta\left(\mathsf{E}[Q_b]\right)}\right)$ for either b if (G, G') is δ-bounded for such a δ.

While δ-boundedness may seem to be a strange condition, we show in Sect. 3.3 that the existing techniques for proving results of the form $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] \leqslant \delta(q_{\mathcal{A}})$ for $\mathcal{A}$ making at most $q_{\mathcal{A}}$ queries can often be easily extended to show the δ-boundedness of a game (G, G'). The examples we consider are the collision-resistance and PRF security of a random oracle and the security of discrete log in the generic group model. In particular, these examples all possess a common form. First, we note that the output of $G(1, \dots)$ is independent of c_0. Consequently, the view of $\mathcal{A}$ when $b = 1$ is independent of c_0 and hence Q_1 is independent of c_0. To analyze $\mathsf{Pr}[\mathsf{BAD} | Q_1 = q]$ we can then think of c' and c_1 being fixed (fixing the transcript of interaction between $\mathcal{A}$ and its oracle in G_1^G) and argue that for any such length q interaction the probability of BAD is bounded by $\delta(q)$ over a random choice of c_0.

We note that this general form seems to typically be implicit in the existing analysis of bad flags for the statistical problems one comes across in ideal model analysis, but would not extend readily to examples where the probability of the

[5] We could simply let $\varepsilon = \Delta / N$ and instead say $\delta(q) = \varepsilon q^d$, but for our examples we found it more evocative to write these terms separately.

bad flag being set is reduced to the probability of an adversary breaking some computational assumption.

3.2 Expected-Time Bound from δ-boundedness

We can now state our result lifting δ-boundedness to a bound on the probability that an adversary sets bad given only its expected number of oracle queries.

Theorem 1. *Let $\delta(q) = \Delta \cdot q^d/N$ for $\Delta, d, N \geqslant 1$. Let (G, G') be a δ-bounded game specification. If $N \geqslant \Delta \cdot 6^d$, then for any $\mathcal{A}$,*

$$\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] \leqslant 5\sqrt[d]{\frac{\Delta \cdot \mathsf{E}[Q_0^{\mathcal{A}}]^d}{N}} = 5\sqrt[d]{\delta\left(\mathsf{E}[Q_0^{\mathcal{A}}]\right)}.$$

If $N \geqslant \Delta \cdot 2^d$, then for any $\mathcal{A}$,

$$\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] \leqslant 3\sqrt[d]{\frac{\Delta \cdot \mathsf{E}[Q_1^{\mathcal{A}}]^d}{N}} = 3\sqrt[d]{\delta\left(\mathsf{E}[Q_1^{\mathcal{A}}]\right)}.$$

We provide bounds based on the expected runtime in either of the two games since they are not necessarily the same. Typically, one of the two games will correspond to a "real" world and it will be natural to desire a bound in terms of the expected runtime in that game. The proof for Q_0 is slightly more complex and is given in this section. The proof for Q_1 is simpler and deferred to the full version of this paper [13]. In the full version we show via a simple attack that the d-th root in these bounds is necessary.

Proof (of Theorem 1). Let $u = 2^{-d}$ and $U = \left\lfloor \sqrt[d]{Nu/\Delta} \right\rfloor$. Note that $\delta(U) \leqslant u$. Now let $\mathcal{B}$ be an adversary that runs exactly like $\mathcal{A}$, except that it counts the number of oracle queries made by $\mathcal{A}$ and halts execution if $\mathcal{A}$ attempts to make a $U+1$-th query. We start our proof by bounding the probability of $\mathsf{BAD}^{\mathcal{A}}$ by the probability of $\mathsf{BAD}^{\mathcal{B}}$ and an $O\left(\sqrt[d]{\delta\left(\mathsf{E}[Q_0^{\mathcal{A}}]\right)}\right)$ term by applying Markov's inequality. In particular we perform the calculations

$$\begin{aligned}
\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}] &= \mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}_{\leqslant U}] + \mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}_{>U}] && (3)\\
&= \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}_{\leqslant U}] + \mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}_{>U}] && (4)\\
&\leqslant \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] + \mathsf{Pr}\left[Q_0^{\mathcal{A}} > U\right] && (5)\\
&\leqslant \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] + \mathsf{E}[Q_0^{\mathcal{A}}]/U && (6)\\
&\leqslant \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] + 3\mathsf{E}[Q_0^{\mathcal{A}}]\sqrt[d]{\Delta/N}. && (7)
\end{aligned}$$

Step 4 follows because for all queries up to the U-th, adversary $\mathcal{B}$ behaves identically to $\mathcal{A}$ (and thus $\mathsf{BAD}_i^{\mathcal{A}} = \mathsf{BAD}_i^{\mathcal{B}}$ for $i \leqslant U$). Step 5 follows because $\mathsf{BAD}^{\mathcal{B}}_{>U}$ cannot occur (because $\mathcal{B}$ never makes more than U queries) and $\mathsf{BAD}^{\mathcal{A}}_{>U}$ can only occur if $Q_0^{\mathcal{A}}$ is at greater than U. Step 6 follows from Markov's inequality.

Step 7 follows from the following calculation which uses the assumption that $N \geqslant \Delta \cdot 6^d$ and that $u = 2^{-d}$,

$$U = \left\lfloor \sqrt[d]{Nu/\Delta} \right\rfloor \geqslant \sqrt[d]{Nu/\Delta} - 1 = \sqrt[d]{N/\Delta}\left(\sqrt[d]{u} - \sqrt[d]{\Delta/N}\right)$$
$$\geqslant \sqrt[d]{N/\Delta}\left(\sqrt[d]{2^{-d}} - \sqrt[d]{\Delta/(\Delta \cdot 6^d)}\right) = \sqrt[d]{N/\Delta}\,(1/2 - 1/6)\,.$$

In the rest of the proof we need to establish that $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] \leqslant 2\mathsf{E}[Q_0^{\mathcal{A}}]\sqrt[d]{\Delta/N}$. We show this with $\mathsf{E}[Q_0^{\mathcal{B}}]$, which is clearly upper bounded by $\mathsf{E}[Q_0^{\mathcal{A}}]$. We will do this by first bounding $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}]$ in terms of $\mathsf{E}[(Q_1^{\mathcal{B}})^d]$, then bounding $\mathsf{E}[(Q_1^{\mathcal{B}})^d]$ in terms of $\mathsf{E}[(Q_0^{\mathcal{B}})^d]$, and then concluding by bounding this in terms of $\mathsf{E}[Q_0^{\mathcal{B}}]$. For the first of these steps we expand $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}]$ by conditioning on all possible values of $Q_1^{\mathcal{B}}$ and applying our assumption that (G, G') is δ-bounded to get

$$\mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] = \sum_{q=1}^{U} \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}} | Q_1^{\mathcal{B}} = q]\mathsf{Pr}[Q_1^{\mathcal{B}} = q] \leqslant \sum_{q=1}^{U} (\Delta \cdot q^d/N)\mathsf{Pr}[Q_1^{\mathcal{B}} = q]$$
$$= \Delta/N \sum_{q=1}^{U} q^d \mathsf{Pr}[Q_1^{\mathcal{B}} = q] = \Delta \mathsf{E}[(Q_1^{\mathcal{B}})^d]/N.$$

So next we will bound $\mathsf{E}[(Q_1^{\mathcal{B}})^d]$ in terms of $\mathsf{E}[(Q_0^{\mathcal{B}})^d]$. To start, we will give a lower bound for $\mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q]$ (when $q \leqslant U$) by using our assumption that (G, G') is δ-bounded. Let R_0 be a random variable which equals $Q_0^{\mathcal{B}}$ if $\mathsf{BAD}^{\mathcal{B}}$ does not occur and equals 0 otherwise. Clearly $R_0 \leqslant Q_0^{\mathcal{B}}$ always. Recall that if $\mathsf{BAD}^{\mathcal{B}}$ does not occur, then $Q_0^{\mathcal{B}} = Q_1^{\mathcal{B}}$ (and hence $R_0 = Q_1^{\mathcal{B}}$) must hold. We obtain

$$\begin{aligned}\mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q] &\geqslant \mathsf{E}[R_0^d | Q_1^{\mathcal{B}} = q] \\ &= q^d \mathsf{Pr}[\neg\mathsf{BAD}^{\mathcal{B}} | Q_1^{\mathcal{B}} = q] + 0^d \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}} | Q_1^{\mathcal{B}} = q] \\ &= q^d(1 - \mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}} | Q_1^{\mathcal{B}} = q]) \\ &\geqslant q^d(1 - \delta(q)) \geqslant q^d(1 - u).\end{aligned}$$

The last step used that $\delta(q) \leqslant \delta(U) \leqslant u$ because $q \leqslant U$.

Now we proceed by expanding $\mathsf{E}[(Q_1^{\mathcal{B}})^d]$ by conditioning on the possible value of $Q_1^{\mathcal{B}}$ and using the above bound to switch $\mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q]$ in for q^d. This gives,

$$\begin{aligned}\mathsf{E}[(Q_1^{\mathcal{B}})^d] &= \sum_{q=1}^{U} q^d \cdot \mathsf{Pr}[Q_1^{\mathcal{B}} = q] \\ &= \sum_{q=1}^{U} \mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q] \cdot \frac{q^d}{\mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q]} \cdot \mathsf{Pr}[Q_1^{\mathcal{B}} = q] \\ &\leqslant \sum_{q=1}^{U} \mathsf{E}[(Q_0^{\mathcal{B}})^d | Q_1^{\mathcal{B}} = q] \cdot \frac{q^d}{q^d(1-u)} \cdot \mathsf{Pr}[Q_1^{\mathcal{B}} = q] \\ &= (1-u)^{-1}\mathsf{E}[(Q_0^{\mathcal{B}})^d]\end{aligned}$$

Our calculations so far give us that $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{B}}] \leqslant (1-u)^{-1}\mathsf{E}[(Q_0^{\mathcal{B}})^d] \cdot \Delta/N$. We need to show that this is bounded by $2\mathsf{E}[Q_0^{\mathcal{B}}] \sqrt[d]{\Delta/N}$. First note that $Q_0^{\mathcal{B}} \leqslant U$ always holds by the definition of $\mathcal{B}$, so

$$(1-u)^{-1}\mathsf{E}[(Q_0^{\mathcal{B}})^d] \cdot \Delta/N \leqslant (1-u)^{-1}\mathsf{E}[Q_0^{\mathcal{B}}] \cdot U^{d-1} \cdot \Delta/N.$$

Now since $U = \left\lfloor \sqrt[d]{Nu/\Delta} \right\rfloor$, we have $U^{d-1} \leqslant (Nu/\Delta)^{(d-1)/d}$ which gives

$$(1-u)^{-1}\mathsf{E}[Q_0^{\mathcal{B}}] \cdot U^{d-1} \cdot \Delta/N \leqslant (1-u)^{-1}(u^{(d-1)/d})\mathsf{E}[Q_0^{\mathcal{B}}] \sqrt[d]{\Delta/N}.$$

Finally, recall that we set $u = 2^{-d}$ and so

$$(1-u)^{-1}(u^{(d-1)/d}) = \frac{2^{-d\cdot(d-1)/d}}{1-2^{-d}} = \frac{2^{1-d}}{1-2^{-d}} \leqslant \frac{2^{1-1}}{1-2^{-1}} = 2.$$

Bounding $\mathsf{E}[Q_0^{\mathcal{B}}] \leqslant \mathsf{E}[Q_0^{\mathcal{A}}]$ and combining with our original bound on $\mathsf{Pr}[\mathsf{BAD}^{\mathcal{A}}]$ completes the proof. □

3.3 Example Applications of Bad Flag Analysis

In this section we walk through some basic examples to show how a bound of $\mathsf{Pr}[\mathsf{bad}|Q_1 = q] \leqslant \Delta \cdot q^d/N$ can be proven using essentially the same techniques as typical bad flag analysis for worst-case runtime, allowing Theorem 1 to be applied. All of our examples follow the basic structure discussed earlier in this section. We write the analysis in terms of two games which are identical-until-bad and parameterized by a bit b. In the $b = 1$ game, the output of its oracles will depend on some coins we identify as c_1, while in the $b = 0$ case the output will depend on both c_1 and independent coins we identify as c_0. Then we think of fixing coins c_1 and the coins used by the adversary, which together fix Q_1 (the number of queries $\mathcal{A}$ would make in the $b = 1$ case), and argue a bound on the probability that bad is set over a random choice of c_0.

We write the necessary games in convenient pseudocode and leave the mapping to a game specification (G, G') to apply Theorem 1 implicit. We will abuse notation and use the name of our pseudocode game to refer to the corresponding game specification.

Collision-resistance of a random oracle. Our first example is the collision resistance of a random oracle. Here an adversary is given access to a random function $h : \{0,1\}^* \to [N]$. It wins if it can find $x \neq y$ for which $h(x) = h(y)$, i.e., a collision in the random oracle. One way to express this is by the game $\mathsf{H}_0^{\mathsf{cr}}$ shown in Fig. 3. The random oracle is represented by the oracle Ro and the oracle Fin allows the adversary to submit supposed collisions.

In it, we have written Ro in a somewhat atypical way to allow comparison to $\mathsf{H}_1^{\mathsf{cr}}$ with which it is identical-until-bad. The coins used by these games determine a permutation π sampled at the beginning of the game and a value X chosen

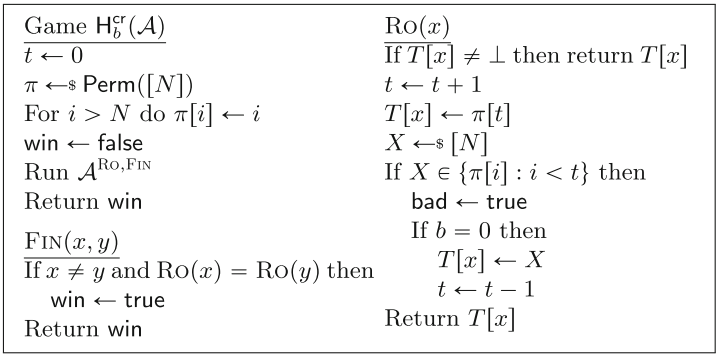

Fig. 3. Game capturing collision-resistance of a random oracle (when $b = 0$).

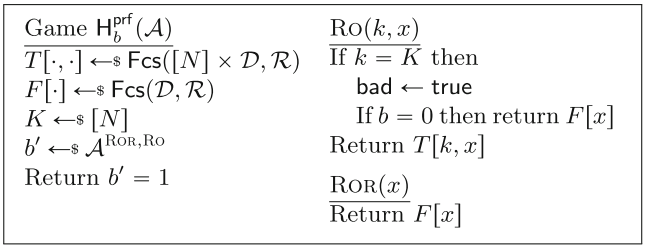

Fig. 4. Games capturing PRF security of a random oracle.

at random from $[N]$ during each RO query[6]. We think of the former as c_1 and the latter as c_0. Ignoring repeat queries, when in $\mathsf{H}_1^{\mathsf{cr}}$ the output of RO is simply $\pi[1], \pi[2], \ldots$ in order. Thus clearly, $\mathsf{Pr}[\mathsf{H}_1^{\mathsf{cr}}(\mathcal{A})] = 0$ since there are no collisions in RO. In $\mathsf{H}_0^{\mathsf{cr}}$ the variable X modifies the output of RO to provide colliding outputs with the correct distribution.

These games are identical-until-bad, so the fundamental lemma of game playing [4] gives us,

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{cr}}(\mathcal{A})] \leqslant \mathsf{Pr}[\mathsf{H}_0^{\mathsf{cr}}(\mathcal{A}) \text{ sets } \mathsf{bad}] + \mathsf{Pr}[\mathsf{H}_1^{\mathsf{cr}}(\mathcal{A})] = \mathsf{Pr}[\mathsf{H}_0^{\mathsf{cr}}(\mathcal{A}) \text{ sets } \mathsf{bad}].$$

Now think of the adversary's coins and the choice of π as fixed. This fixes a value of Q_1 and a length Q_1 transcript of $\mathcal{A}$'s queries in $\mathsf{H}_1^{\mathsf{cr}}(\mathcal{A})$. If $\mathcal{A}$ made all of its queries to FIN, then RO will have been executed $2Q_1$ times. On the i-th query to RO, there is at most an $(i-1)/N$ probability that the choice of X will cause bad to be set. By a simple union bound we can get,

$$\mathsf{Pr}[\mathsf{BAD} | Q_1 = q] \leqslant q(2q-1)/N.$$

[6] We define $\pi[i] = i$ for $i > N$ just so the game $\mathsf{H}_1^{\mathsf{cr}}$ is well-defined if $\mathcal{A}$ makes more than N queries.

Setting $\delta(q) = 2q^2/N$ we have that H^{cr} is δ-bounded, so Theorem 1 gives

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{cr}}(\mathcal{A})] \leqslant 5\sqrt[2]{\frac{2 \cdot \mathsf{E}[Q_0^{\mathcal{A}}]^2}{N}}.$$

Pseudorandomness of a random oracle. Now consider using a random oracle with domain $[N] \times \mathcal{D}$ and range $\mathcal{R}$ as a pseudorandom function. The games for this are shown in Fig. 4. The real world is captured by $b = 0$ (because to output of the random oracle Ro is made to be consistent with output of the real-or-random oracle Ror) and the ideal world by $b = 1$.

The coins of the game are random tables T and F as well as a random key K. We think of the key as c_0 and the tables as c_1. Because we have written the games so that the consistency check occurs in Ro, we can clearly see the output of the oracles in $\mathsf{H}_1^{\mathsf{prf}}$ are independent of $c_0 = K$.

These games are identical-until-bad so from the fundamental lemma of game playing we have,

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{prf}}(\mathcal{A})] - \mathsf{Pr}[\mathsf{H}_1^{\mathsf{prf}}(\mathcal{A})] \leqslant \mathsf{Pr}[\mathsf{H}_0^{\mathsf{prf}}(\mathcal{A}) \text{ sets } \mathsf{bad}].$$

Now we think of c_1 and the coins of $\mathcal{A}$ as fixed. Over a random choice of K, each Ro query has a $1/N$ change of setting bad. By a simple union bound we get,

$$\mathsf{Pr}[\mathsf{BAD} | Q_1 = q] \leqslant q/N.$$

Defining $\delta(q) = q/N$ we have that $\mathsf{H}^{\mathsf{prf}}$ is δ-bounded, so Theorem 1 gives

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{prf}}(\mathcal{A})] - \mathsf{Pr}[\mathsf{H}_1^{\mathsf{prf}}(\mathcal{A})] \leqslant 5 \cdot \mathsf{E}[Q_0^{\mathcal{A}}]/N.$$

Discrete logarithm security in the generic group model. Next we consider discrete logarithm security in the generic group model for a prime order group $\mathbb{G}$ with generator g. One way to express this is by the game $\mathsf{H}_0^{\mathsf{dl}}$ shown in Fig. 5. In this expression, the adversary is given labels for the group elements it handles based on the time that this group element was generated by the adversary. The more general framing of the generic group model where $g^x \in \mathbb{G}$ is labeled by $\sigma(x)$ for a randomly chosen $\sigma : \mathbb{Z}_{|\mathbb{G}|} \to \{0,1\}^l$ for some $l \geqslant \lceil \log |\mathbb{G}| \rceil$ can easily be reduced to this version of the game.

At the beginning of the game polynomials $p_0(\cdot) = 0$, $p_1(\cdot) = 1$, and $p_2(\cdot) = X$ are defined. These are polynomials of the symbolic variable X, defined over $\mathbb{Z}_{|\mathbb{G}|}$. Then a random x is sampled and the goal of the adversary is to find this x. Throughout the game, a polynomial p_i represents the group element $g^{p_i(x)}$. Hence p_0 represents the identity element of the group, p_1 represents the generator g, and p_2 represents g^x. We think of the subscript of a polynomial as the adversary's label for the corresponding group element. The variable t tracks the highest label the adversary has used so far.

We let $\mathcal{P}^i$ denote the set of the first i polynomials that have been generated and $\mathcal{P}_x^i$ be the set of their outputs when evaluated on x. The oracle Init tells the adversary if x happened to be 0 or 1 by returning the appropriate value

Game $\mathsf{H}_b^{\mathsf{dl}}(\mathcal{A})$	$\textsc{Op}(\boldsymbol{j}, \boldsymbol{\alpha})$
$p_0(\cdot) \leftarrow 0$; $p_1(\cdot) \leftarrow 1$	Require $\boldsymbol{j}[i] \leqslant t$ for $i = 1, \ldots, \lvert\boldsymbol{j}\rvert$
$p_2(\cdot) \leftarrow X$	Require $\boldsymbol{\alpha} \in \mathbb{Z}_{\lvert\mathbb{G}\rvert}^{\lvert\boldsymbol{j}\rvert}$
$t \leftarrow 2$; $x \leftarrow\!\!\$\ \mathbb{Z}_{\lvert\mathbb{G}\rvert}$	$t \leftarrow t + 1$
$x' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Init},\textsc{Op}}$	$p_t(\cdot) \leftarrow \sum_{i=1}^{\lvert\boldsymbol{j}\rvert} \boldsymbol{\alpha}[i] \cdot p_{\boldsymbol{j}[i]}(\cdot)$
Return $x = x'$	$\ell \leftarrow t$
$\textsc{Init}()$	If $p_t(\cdot) \in \mathcal{P}^{t-1}$ then
$\ell \leftarrow 2$	$\quad \ell \leftarrow \min\{k < t : p_t(\cdot) = p_k(\cdot)\}$
If $p_2(x) \in \mathcal{P}_x^1$ then	If $p_t(x) \in \mathcal{P}_x^{t-1}$ and $p_t(\cdot) \notin \mathcal{P}^{t-1}$ then
$\quad \mathsf{bad} \leftarrow \mathsf{true}$	$\quad \mathsf{bad} \leftarrow \mathsf{true}$
$\quad$ If $b = 0$ then $\ell \leftarrow x$	$\quad$ If $b = 0$ then $\ell \leftarrow \min\{k < t : p_t(x) = p_k(x)\}$
Return ℓ	Return ℓ

Fig. 5. Game capturing discrete logarithm security of a generic group (when $b = 0$). For $i \in \mathbb{N}$ and $x \in \mathbb{Z}_{|\mathbb{G}|}$, we use the notation $\mathcal{P}^i = \{p_0, \ldots, p_i\} \subset \mathbb{Z}_{|\mathbb{G}|}[X]$ and $\mathcal{P}_x^i = \{p(x) : p \in \mathcal{P}^i\} \subset \mathbb{Z}_{|\mathbb{G}|}$.

of ℓ. The oracle Op allows the adversary to perform multi-exponentiations. It specifies a vector $\boldsymbol{j}$ of labels for group elements and a vector $\boldsymbol{\alpha}$ of coefficients. The variable t is incremented and its new value serves as the label for the group element $\prod_i g_{\boldsymbol{j}[i]}^{\boldsymbol{\alpha}[i]}$ where $g_{\boldsymbol{j}[i]}$ is the group element with label $\boldsymbol{j}[i]$, i.e., $g^{p_{\boldsymbol{j}[i]}(x)}$. The returned value ℓ is set equal to the prior label of a group element which equals this new group element (if $\ell = t$, then no prior labels represented the same group element).

The only coins of this game are the choice of x which we think of as c_0. In $\mathsf{H}_1^{\mathsf{dl}}$, the adversary is never told when two labels it handles non-trivially represent the same group element so the view of $\mathcal{A}$ is independent of c_0, as desired[7]. Because the view of $\mathcal{A}$ is independent of x when $b = 1$ we have that $\mathsf{Pr}[\mathsf{H}_1^{\mathsf{dl}}(\mathcal{A})] = 1/|\mathbb{G}|$.

From the fundamental lemma of game playing,

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{dl}}(\mathcal{A})] \leqslant \mathsf{Pr}[\mathsf{H}_0^{\mathsf{dl}}(\mathcal{A}) \text{ sets } \mathsf{bad}] + \mathsf{Pr}[\mathsf{H}_1^{\mathsf{dl}}(\mathcal{A})] = \mathsf{Pr}[\mathsf{H}_0^{\mathsf{cr}}(\mathcal{A}) \text{ sets } \mathsf{bad}] + 1/|\mathbb{G}|$$

Now thinking of the coins of $\mathcal{A}$ as fixed, this fixes a value of Q_1 and a length Q_1 transcript of queries that would occur in $\mathsf{H}_1^{\mathsf{dl}}(\mathcal{A})$. This in turn fixes the set of polynomials $\mathcal{P}^{Q_1+2}$. The flag bad will be set iff any of polynomials in the set

$$\{p(\cdot) - r(\cdot) | p \neq r \in \mathcal{P}^{Q_1+2}\}$$

have the value 0 when evaluated on x. Note these polynomials are non-zero and have degree at most 1. Thus, applying the Schwartz-Zippel lemma and a union bound we get,

$$\mathsf{Pr}[\mathsf{BAD}|Q_1 = q] \leqslant \binom{q+3}{2} \cdot (1/|\mathbb{G}|) \leqslant 6q^2/|\mathbb{G}|.$$

[7] Two labels trivially represent the same group element if they correspond to identical polynomials.

Note the bound trivially holds when $q = 0$, since $\mathsf{Pr}[\mathsf{bad}|Q_1 = q] = 0$, so we have assumed $q \geqslant 1$ for the second bound. Defining $\delta(q) = 6q^2/|\mathbb{G}|$ we have that H^{dl} is δ-bounded, so Theorem 1 gives

$$\mathsf{Pr}[\mathsf{H}_0^{\mathsf{dl}}(\mathcal{A})] \leqslant 5\sqrt[2]{\frac{6 \cdot \mathsf{E}[Q_0^{\mathcal{A}}]^2}{|\mathbb{G}|}} + \frac{1}{|\mathbb{G}|}.$$

4 Concrete Security for a Forking Lemma

In this section we apply our techniques to obtaining concrete bounds on the soundness of proof systems. Of particular interest to us will be proof systems that can be proven to achieve a security notion known as witness-extended emulation via a very general "Forking Lemma" introduced by Bootle, Cerulli, Chaidos, Groth, and Petit (BCCGP) [6]. Some examples include Bulletproofs [7], Hyrax [24], and Supersonic [8]. Our expected-time techniques arise naturally for these proof systems because witness-extended emulation requires the existence of an expected-time *emulator* E for a proof system which is given oracle access to a cheating prover and produces transcripts with the same distribution as the cheating prover, but additionally provides a witness w for the statement being proven whenever it outputs an accepting transcript.

In this section we use a new way of expressing witness-extended emulation as a special case of a more general notion we call *predicate-extended emulation*. The more general notion will serve as a clean, modular way to provide a concrete security version of the BCCGP forking lemma. This modularity allows us to hone in on the steps where our expected time analysis can be applied to give concrete bounds and avoid some technical issues with the original BCCGP formulation of the lemma.

In the BCCGP blueprint, the task of witness-extended emulation is divided into a generic *tree-extended emulator* which for any public coin proof system produces transcripts with the same distribution as a cheating prover together with a set of accepting transcripts satisfying a certain tree structure and an *extractor* for the particular proof system under consideration which can extract a witness from such a tree of transcripts. The original forking lemma of BCCGP technically only applied for extractors that always output a witness given a valid tree with no collisions. However, typical applications of the lemma require that the extractor be allowed to fail when the cheating prover has (implicitly) broken some presumed hard computational problem. Several works subsequent to BCCGP noticed this gap in the formalism [7,8,24] and stated slight variants of the BCCGP forking lemma. However, these variants are still unsatisfactory. The variant lemmas in [7,24] technically only allows extractors which fail in extracting a witness with at most negligible probability *for every tree* (rather than negligible probably with respect to some efficiently samplable distribution over trees, as is needed). The more recent variant lemma in [8] is stated in such a way that the rewinding analysis at the core of the BCCGP lemma is *omitted* from the variant lemma and (technically) must be shown separately anytime it

Π	$\Pi(\pi, u, \text{aux})$ returns true iff	Name
Π_{wit}	$(u, \text{aux}) \in \mathsf{R}_\pi$	Valid Witness
$\Pi_{\mathsf{dl}}^{\mathbb{G},n}$	$\prod_{i=0}^{n} \pi_i^{\text{aux}_i} = 1_{\mathbb{G}}$	Discrete Log Relation
$\Pi_{\mathsf{val}}^{\boldsymbol{n}}$	aux is a valid $\boldsymbol{n}$-tree for u	Valid Tree
$\Pi_{\mathsf{nocol}}^{\boldsymbol{n}}$	aux has no challenge collisions	Collision-Free Tree

Fig. 6. Predicates we use. Other predicates Π_{bind} and Π_{rsa} are only discussed informally.

is to be applied to a proof system. None of these issues represent issues with the security of the protocols analyzed in these works. The intended meaning of each of their proofs is clear from context and sound, these issues are just technical bugs with the formalism of the proofs. However, to accurately capture concrete security it will be important that we have a precise and accurate formalism of this. Our notion of predicate-extended emulation helps to enable this.

In Sect. 4.1, we provide the syntax of proof systems as well as defining our security goals of predicate-extended emulation (a generalization of witness-extended emulation) and generator soundness (a generalization of the standard notion of soundness). Then in Sect. 4.2, we provide a sequence of simple lemmas and show how they can be combined to give our concrete security version on the forking lemma. Finally in Sect. 4.3, we discuss how our forking lemma can easily be applied to provide concrete bounds on the soundness of various existing proof systems. As a concrete example we give the first concrete security bound on the soundness of the Bulletproof zero-knowledge proof system for arithmetic circuits by Bünz et al. [7].

4.1 Syntax and Security of Proof Systems

Proof System. A proof system PS is a tuple $\mathsf{PS} = (\mathsf{S}, \mathsf{R}, \mathsf{P}, \mathsf{V}, \mu)$ specifying a setup algorithm S, a relation R, a prover P, verifier V, and $\mu \in \mathbb{N}$. The setup algorithm outputs public parameters π. We say w is a witness for the statement u if $(u, w) \in \mathsf{R}_\pi$. The prover (with input (u, w)) and the verifier (with input u) interact via $2\mu + 1$ moves as shown in Fig. 7.

Here tr is the *transcript* of the interaction and $d \in \{0, 1\}$ is the *decision* of V (with $d = 1$ representing acceptance and $d = 0$ representing rejection). *Perfect completeness* requires that for all π and $(u, w) \in \mathsf{R}_\pi$, $\mathsf{Pr}[d = 1 : (\cdot, d) \leftarrow\!\!\$\ \langle \mathsf{P}_\pi(u, w), \mathsf{V}_\pi(u) \rangle] = 1$. If PS is public-coin, then m_{2i-1} output by V each round is set equal to its random coins. In this case, we let

$\underline{\langle \mathsf{P}_\pi(u, w), \mathsf{V}_\pi(u) \rangle}$
$\sigma_\mathsf{P} \leftarrow \bot$; $\sigma_\mathsf{V} \leftarrow \bot$; $m_{-1} \leftarrow \bot$
$(m_0, \sigma_\mathsf{P}) \leftarrow\!\!\$\ \mathsf{P}_\pi(u, w, m_{-1}, \sigma_\mathsf{P})$
For $i = 1, \ldots, \mu$ do
  $(m_{2i-1}, \sigma_\mathsf{V}) \leftarrow\!\!\$\ \mathsf{V}_\pi(u, m_{2i-2}, \sigma_\mathsf{V})$
  $(m_{2i}, \sigma_\mathsf{P}) \leftarrow\!\!\$\ \mathsf{P}_\pi(u, w, m_{2i-1}, \sigma_\mathsf{P})$
$tr \leftarrow (m_{-1}, m_0, m_1, \ldots, m_{2\mu})$
$d \leftarrow \mathsf{V}_\pi(m_{2\mu}, u, \sigma_\mathsf{V})$
Return (tr, d)

Fig. 7. Interaction between (honest) prover P and verifier V with public parameters π. Here tr is the transcript and $d \in \{0, 1\}$ is the decision.

$\mathsf{V}_\pi(u, tr) \in \{0,1\}$ denote V's decision after an interaction that produced transcript tr[8]. Throughout this section we will implicitly assume that any proof systems under discussion is public-coin. We sometimes refer to the verifier's outputs as challenges.

Predicate-extended emulation. The proof systems we consider were all analyzed with the notion of *witness-extended emulation* [11,16]. This requires that for any efficient cheating prover P^* there exists an efficient emulator E which (given oracle access to P^* interacting with V and the ability to rewind them) produces transcripts with the same distribution as P^* and almost always provides a witness for the statement when the transcript it produces is accepting. We will capture witness-extended emulation as a special case of what we refer to as *predicate-extended emulation*. We cast the definition as two separate security properties. The first (emulation security) requires that E produces transcripts with the same distribution as P^*. The second (predicate extension) is parameterized by a predicate Π and requires that whenever E produces an accepting transcript, its auxiliary output must satisfy Π. As we will see, this treatment will allow a clean, modular treatment of how BCCGP and follow-up work [6–8,24] analyze witness-extended emulation.

We start by considering game $\mathsf{H}^{\mathsf{emu}}$ defined in Fig. 8. It is parameterized by a public-coin proof system PS, emulator E, and bit b. The adversary consists of a cheating prover P^* and an attacker $\mathcal{A}$. This game measures $\mathcal{A}$'s ability to distinguish between a transcript generated by $\langle \mathsf{P}^*_\pi(u,s), \mathsf{V}_\pi(u)\rangle$ and one generated by E. The emulator E is given access to oracles Next and Rew. The former has P^* and V perform a round of interaction and returns the messages exchanged. The latter rewinds the interaction to the prior round. We define the advantage function $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}}(\mathsf{P}^*,\mathcal{A}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E},1}(\mathsf{P}^*,\mathcal{A})] - \mathsf{Pr}[\mathsf{H}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E},0}(\mathsf{P}^*,\mathcal{A})]$. For the examples we consider there will be an E which (in expectation) performs a small number of oracle queries and does a small amount of local computation such that for any P^* and $\mathcal{A}$ we have $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}}(\mathsf{P}^*,\mathcal{A}) = 0$.

Note that creating a perfect emulator is trivial in isolation; E can just make $\mu+1$ calls to Next to obtain a tr with the exactly correct distribution. Where it gets interesting is that we will consider a second, auxiliary output of E and insist that it satisfies some predicate Π whenever tr is an accepting transcript. The adversary wins whenever tr is accepting, but the predicate is not satisfied. This is captured by the game $\mathsf{H}^{\mathsf{predext}}$ shown in Fig. 8. We define $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi}(\mathsf{P}^*,\mathcal{A}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi}(\mathsf{P}^*,\mathcal{A})]$. Again this notion is trivial in isolation; E can just output rejecting transcripts. Hence, both security notions need to be considered together with respect to the same E.

The standard notion of witness-extended emulating is captured by the predicate Π_{wit} which checks if aux is a witness for u, that is, $\Pi_{\mathsf{wit}}(\pi, u, \text{aux}) = ((u, \text{aux}) \in \mathsf{R}_\pi)$. Later we will define some other predicates. All the predicates we will make use of are summarized in Fig. 6. A proof system with a good witness-

[8] We include $m_{-1} = \perp$ in tr as a notational convenience.

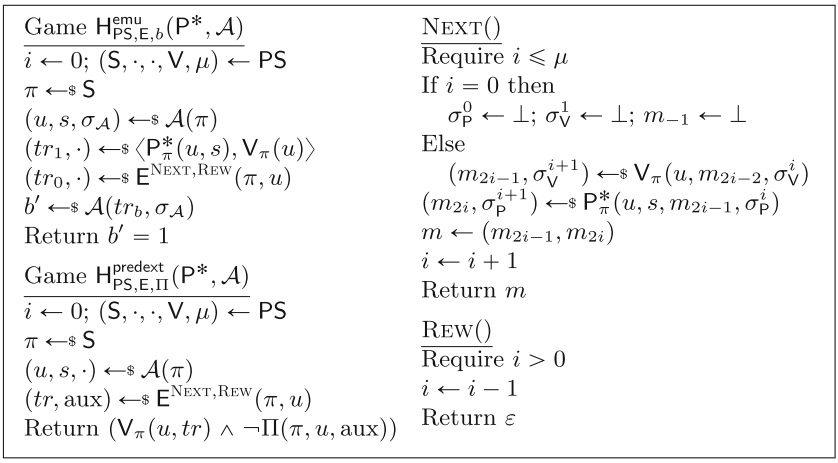

Fig. 8. Games defining predicate-extended emulation security of proof system PS.

extended emulator under some computational assumption may be said to be an *argument of knowledge*.

Hard predicates. One class of predicates to consider are those which embed some computational problem about the public parameter π that is assumed to be hard to solve. We will say that Π is *witness-independent* if its output does not depend on its second input u. For example, if S outputs of length n vector of elements from a group $\mathbb{G}$ (we will denote this setup algorithm by $\mathsf{S}^n_{\mathbb{G}}$) we can consider the predicate $\Pi^{\mathbb{G},n}_{\mathsf{dl}}$ which checks if aux specifies a non-trivial discrete log relation. This predicate is useful for the analysis of a variety of proof systems [6,7,24]. Other useful examples include: (i) if S output parameters for a commitment scheme with Π_{bind} that checks if aux specifies a commitment and two different opening for it [6,8,24] and (ii) if S outputs a group of unknown order together with an element of that group and Π_{rsa} checks if aux specifies a non-trivial root of that element [8].

Whether a witness-independent predicate Π is hard to satisfy given the output of S is captured by the game $\mathsf{H}^{\mathsf{pred}}$ shown on the left side of Fig. 9. We define $\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{S},\Pi}(\mathcal{A}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{pred}}_{\mathsf{S},\Pi}(\mathcal{A})]$. Note, for example, that if $\mathsf{S}^n_{\mathbb{G}}$ and $\Pi^{\mathbb{G},n}_{\mathsf{dl}}$ is used, then this game is identical to discrete log relation security, i.e., $\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{S}^n_{\mathbb{G}},\Pi^{\mathbb{G},n}_{\mathsf{dl}}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{dl\text{-}rel}}_{\mathbb{G},n}(\mathcal{A})$ for any adversary $\mathcal{A}$.

Generator soundness. Consider the games shown on the right side of Fig. 9. Both are parameterized by a *statement generator* G which (given the parameters π) outputs a statement u and some auxiliary information s about the statement. The first game $\mathsf{H}^{\mathsf{sound}}$ measure how well a (potentially cheating) prover P^* can use s to convince V that u is true. The second game $\mathsf{H}^{\mathsf{wit}}$ measures how well

Game $\mathsf{H}^{\mathsf{pred}}_{\mathsf{S},\Pi}(\mathcal{A})$	Game $\mathsf{H}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*)$	Game $\mathsf{H}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B})$
$\pi \leftarrow\!\!\$\ \mathsf{S}$	$(\mathsf{S},\cdot,\cdot,\mathsf{V},\cdot) \leftarrow \mathsf{PS}$	$(\mathsf{S},\cdot,\cdot,\mathsf{V},\cdot) \leftarrow \mathsf{PS}$
$\mathrm{aux} \leftarrow\!\!\$\ \mathcal{A}(\pi)$	$\pi \leftarrow\!\!\$\ \mathsf{S}$	$\pi \leftarrow\!\!\$\ \mathsf{S}$
Return $\Pi(\pi,\varepsilon,\mathrm{aux})$	$(u,s) \leftarrow\!\!\$\ G(\pi)$	$(u,s) \leftarrow\!\!\$\ G(\pi)$
	$(\cdot,d) \leftarrow\!\!\$\ \langle \mathsf{P}^*_\pi(u,s), \mathsf{V}_\pi(u)\rangle$	$w \leftarrow \mathcal{B}(\pi,u,s)$
	Return $d=1$	Return $(u,w) \in \mathsf{R}_\pi$

Fig. 9. Left. Game defining hardness of satisfying predicate Π. **Right.** Games defining soundness of proof system PS with respect to instance generator G and difficulty of finding witness for statements produced by G.

an adversary $\mathcal{B}$ can produce a witness for u given s. We define $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*) = \mathsf{Pr}[\mathsf{H}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*)]$ and $\mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B}) = \mathsf{Pr}[\mathsf{H}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B})]$.

Note that the standard notion of soundness (that proving false statements is difficult) is captured by considering G which always outputs false statements. In this case, $\mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{A}) = 0$ for all $\mathcal{A}$. In other contexts, it may be assumed that it is computationally difficult to find a witness for G's statement.

4.2 Concrete Security Forking Lemma

Now we will work towards proving our concrete security version of the BCCGP forking lemma. This lemma provides a general framework for how to provide a good witness-extended emulator for a proof system. First, BCCGP showed how to construct a tree-extended emulator T which has perfect emulation security and (with high probability) outputs a set of transcripts satisfying a tree-like structure (defined later) whenever it outputs an accepting transcript. Then one constructs, for the particular proof system under consideration, an "extractor" X which given such a tree of transcripts can always produce a witness for the statement or break some other computational problem assumed to be difficult. Combining T and X appropriately gives a good witness-extended emulator.

Before proceeding to our forking lemma we will provide the necessary definitions of a tree-extended emulator and extractor, then state some simple lemmas that help build toward our forking lemma.

Transcript Tree. Fix a proof system $\mathsf{PS} = (\mathsf{S},\mathsf{R},\mathsf{P},\mathsf{V},\mu)$ and let the vector $\boldsymbol{n} = (n_1,\ldots,n_\mu) \in \mathbb{N}^\mu_{>0}$ be given. Let π be an output of S and u be a statement. For $h = 0,\ldots,\mu$ we will inductively define an $(n_{\mu-h+1},\ldots,n_\mu)$-tree of transcripts for (PS,π,u). We will often leave some of (PS,π,u) implicit when they are clear from context.

First when $h = 0$, a ()-tree is specified by a tuple $(m_{2\mu-1}, m_{2\mu}, \ell)$ where $m_{2\mu-1}, m_{2\mu} \in \{0,1\}^*$ and ℓ is an empty list. Now an $(n_{\mu-(h+1)},\ldots,n_\mu)$-tree is specified by a tuple $(m_{2(\mu-h)-1}, m_{2(\mu-h)}, \ell)$ where $m_{2(\mu-h)-1}, m_{2(\mu-h)} \in \{0,1\}^*$ and ℓ is a length $n_{\mu-(h+1)}$ list of $(n_{\mu-h},\ldots,n_\mu)$-trees for (PS,π,u,tr).

When discussing such trees we say their height is h. When $h < \mu$ we will sometimes refer to it as a *partial tree*. We use the traditional terminology of

nodes, *children*, *parent*, *root*, and *leaf*. We say the root node is at height h, its children are at height $h-1$, and so on. The leaf nodes are thus each at height 0. If a node is at height h, then we say it is at depth $\mu - h$.

Every path from the root to a leaf in a height h tree gives a sequence $(m_{2(\mu-h)-1}, m_{2(\mu-h)}, \ldots, m_{2\mu-1}, m_{2\mu})$ where $(m_{2(\mu-i)-1}, m_{2(\mu-i)})$ are the pair from the node at height i. Now if we fix a transcript prefix $tr' = (m_{-1}, m_0, \ldots, m_{2(\mu-h-1)-1}, m_{2(\mu-h-1)})$, then we can think of tr' and the tree as inducing $\prod_{i=1}^{\mu} n_i$ different transcripts $tr = (m_0, \ldots, m_{2\mu-1}, m_{2\mu})$, one for each path. We will say that the tree is *valid* for tr' if $\mathsf{V}_\pi(u, tr) = 1$ for each transcript tr induced by the tree. Note that tr' is an empty list when $h = \mu$ so we can omit reference to tr' and simply refer to the tree as valid.

Suppose V's coins are drawn from $S \times \mathbb{Z}_p$ for some set S and $p \in \mathbb{N}$. We will refer to the second component of its coins are the integer component. Let node be a parent node at height $i > 0$. If any two of its children have $m_{2(\mu-i+1)-1}$ with identical integer components, then we say that node has a challenge collision. A tree has a *challenge collision* if any of its nodes have a challenge collision.

A tree-extractor emulator should return trees which are valid and have no challenge collision. We capture this with the predicates $\Pi^{\boldsymbol{n}}_{\mathsf{val}}$ and $\Pi^{\boldsymbol{n}}_{\mathsf{nocol}}$ defined by:

- $\Pi^{\boldsymbol{n}}_{\mathsf{val}}(\pi, u, \mathrm{aux})$ returns true iff aux is a valid $\boldsymbol{n}$-tree.
- $\Pi^{\boldsymbol{n}}_{\mathsf{nocol}}(\pi, u, \mathrm{aux})$ returns true iff aux is an $\boldsymbol{n}$-tree that does not have a challenge collision.

Tree-extended Emulator. Let a proof system $\mathsf{PS} = (\mathsf{S}, \mathsf{R}, \mathsf{P}, \mathsf{V}, \mu)$ and let $(n_1, \ldots, n_\mu) \in \mathbb{N}^{\mu}_{>2}$ be given. Then consider the tree-extended emulator T given in Fig. 10 which comes from BCCGP. The sub-algorithms T_i are given a partial transcript tr. They call Next to obtain the next messages of a longer partial transcript and attempt to create a partial tree with is valid for it. This is done by repeatedly calling T_{i+1} to construct each branch of the tree. Should the first such call fail, then T_i will abort. Otherwise, it will continue calling T_{i+1} as many times as necessary to have n_{i+1} branches. The base case of this process is T_μ which does not need children branches and instead just checks if its transcript is accepting, returning $\perp$ to its calling procedure if not. The following result shows that T emulates any cheating prover perfectly and almost always outputs a valid tree whenever it outputs an accepting transcript. The technical core of the lemma is in the bound on the expected efficiency of T.

Lemma 4. *Let $\mathsf{PS} = (\mathsf{S}, \mathsf{R}, \mathsf{P}, \mathsf{V}, \mu)$ be a public coin proof system. Suppose V's challenges are uniformly drawn from $S \times \mathbb{Z}_p$ for set S and $p \in \mathbb{N}$. Let $\boldsymbol{n} = (n_1, \ldots, n_\mu) \in \mathbb{N}^{\mu}_{>0}$ be given. Let $N = \prod_{i=1}^{\mu} n_i$. Let P^* be a cheating prover and $\mathcal{A}$ be an adversary. Define T as shown in Fig. 10. Then the following all hold:*

1. $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{T}}(\mathsf{P}^*, \mathcal{A}) = 0$
2. $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{T},\Pi^{\boldsymbol{n}}_{\mathsf{val}}}(\mathsf{P}^*, \mathcal{A}) = 0$
3. $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{T},\Pi^{\boldsymbol{n}}_{\mathsf{nocol}}}(\mathsf{P}^*, \mathcal{A}) \leqslant 5\mu N/\sqrt{2p}$
4. *The expected number of times T executes $\mathsf{V}_\pi(u, \cdot)$ is N.*

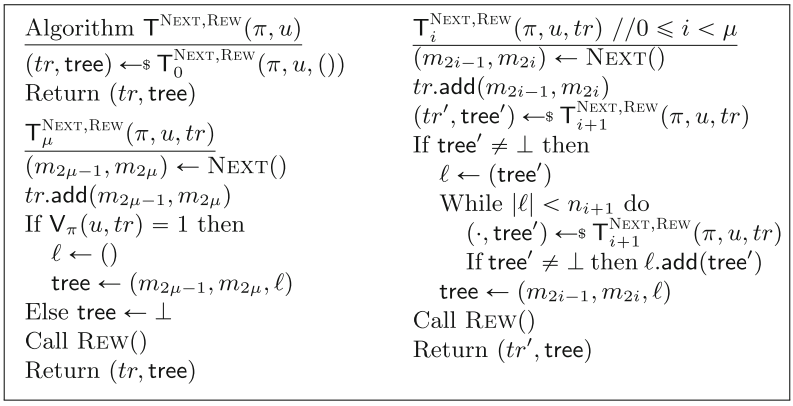

Fig. 10. The BCCGP tree-extended emulator.

5. *The expected number of queries that* T *makes to* Next *is less than* $\mu N + 1$[9]. *Exactly one of these queries is made while* $i = 1$ *in* Next.

For comparison, in the full version of this paper [13] we analyze a natural tree-extended emulator with a small bounded worst-case runtime. Its ability to produce valid trees is significantly reduced by its need to work within a small worst-case runtime, motivating the need for T to only be efficient in expected runtime.

Proof (of Lemma 4). All of the claims except the third follow from BCCGP's analysis of T. The advantage $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{T},\Pi^n_{\mathsf{nocol}}}(\mathsf{P}^*, \mathcal{A})$ can be upper-bounded by the probability that the integer component of V's output is repeated across *any* of T's queries to Next. BCCGP bounded this probability by applying Markov's inequality to obtain an upper bound on T's running time and then applying the birthday bound to get an $O(\mu N/\sqrt[3]{p})$ bound. We can instead apply our switching lemma analysis from the full version of this paper [13] (or the techniques from our analysis of the collision resistance of a random oracle in Sect. 3.3) to obtain the stated bound because V will sample μN challenges in expectation. □

Extractors. Let X be an algorithm and Π_1, Π_2 be predicates. We say that X is a (Π_1, Π_2)-extractor if $\Pi_1(\pi, u, \text{aux}) \Rightarrow \Pi_2(\pi, u, \mathsf{X}(\pi, u, \text{aux}))$. Let T be an emulator. Then we define $\mathsf{E}^\dagger[\mathsf{T}, \mathsf{X}]$ to be the emulator that on input (π, u) with oracle access to Next and Rew will first compute $(tr, \text{aux}) \leftarrow\!\!\$\ \mathsf{T}^{\text{Next,Rew}}(\pi, u)$ and then returns $(tr, \mathsf{X}(\pi, u, \text{aux}))$. The following straightforward lemma relates the security of T and $\mathsf{E}^\dagger$.

Lemma 5. *Let* PS *be a proof system,* T *be an emulator,* Π_1 *and* Π_2 *be predicates,* P^* *be a cheating prover, and* $\mathcal{A}$ *be an adversary. Let* X *be a* (Π_1, Π_2)*-extractor. Then the following hold:*

[9] More precisely, the expected number of queries that T makes to Next is the number of nodes in a $(n_1, \ldots, n_\mu)$-tree. This is $\sum_{i=0}^{\mu} \prod_{j=1}^{i} n_j$, where $\prod_{j=1}^{0} n_j = 1$.

Adversary $\mathcal{B}_{\mathsf{E}}(\pi)$
$i \leftarrow 0$
$(u, s, \cdot) \leftarrow\!\!\$\ \mathcal{A}(\pi)$
$(\cdot, \text{aux}) \leftarrow\!\!\$\ \mathsf{E}^{\text{Next,Rew}}(\pi, u)$
Return aux

Next,Rew
//Defined as in Fig. 8

Fig. 11. Reduction adversary for Theorem 2.

- $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}^\dagger[\mathsf{T},\mathsf{X}]}(\mathsf{P}^*,\mathcal{A}) = \mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{T}}(\mathsf{P}^*,\mathcal{A})$
- $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E}^\dagger[\mathsf{T},\mathsf{X}],\Pi_2}(\mathsf{P}^*,\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{T},\Pi_1}(\mathsf{P}^*,\mathcal{A})$

Forking lemma. Finally, we can state and prove our concrete security version of the BCCGP forking lemma. It captures the fact that any protocol with a $(\Pi^{\boldsymbol{n}}_{\mathsf{val}} \wedge \Pi^{\boldsymbol{n}}_{\mathsf{nocol}}, \Pi_{\mathsf{wit}} \vee \Pi^*)$-extractor has a good witness-extended emulator (assuming Π^* is computationally difficult to satisfy)[10].

Theorem 2 (Forking Lemma). *Let $\mathsf{PS} = (\mathsf{S},\mathsf{R},\mathsf{P},\mathsf{V},\mu)$ be a public coin proof system. Suppose V's challenges are uniformly drawn from $S \times \mathbb{Z}_p$ for set S and $p \in \mathbb{N}$. Let $\boldsymbol{n} = (n_1,\ldots,n_\mu) \in \mathbb{N}^\mu_{>0}$ be given. Let $N = \prod_{i=1}^{\mu} n_i$. Let P^* be a cheating prover and $\mathcal{A}$ be an adversary. Define T as shown in Fig. 10. Let Π^* be a witness-independent predicate. Let X be a $(\Pi^{\boldsymbol{n}}_{\mathsf{val}} \wedge \Pi^{\boldsymbol{n}}_{\mathsf{nocol}}, \Pi_{\mathsf{wit}} \vee \Pi^*)$-extractor. Let $\mathsf{E} = \mathsf{E}^\dagger[\mathsf{T},\mathsf{X}]$. Let $\mathcal{B}_{\mathsf{E}}$ be as defined in Fig. 11. Then the following all hold:*

1. $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}}(\mathsf{P}^*,\mathcal{A}) = 0$
2. $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_{\mathsf{wit}}}(\mathsf{P}^*,\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PS},\Pi^*}(\mathcal{B}_{\mathsf{E}}) + 5\mu N/\sqrt{2p}$
3. *The expected number of times T executes $\mathsf{V}_\pi(u,\cdot)$ (inside of E) is N.*
4. *The expected number of queries that E makes to* Next *is less than $\mu N + 1$. Exactly one of these queries is made while $i = 1$ in* Next.
5. *The expected runtime of $\mathcal{B}_{\mathsf{E}}$ is approximately $T_{\mathcal{A}} + Q_{\mathsf{E}} \cdot T_{\mathsf{P}^*} + T_{\mathsf{E}}$ where T_x is the worst-case runtime of $x \in \{\mathcal{A},\mathsf{P}^*,\mathsf{E}\}$ and $Q_{\mathsf{E}} < \mu N + 1$ is the expected number of queries that E makes to* Next *in $\mathsf{H}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi^*}(\mathsf{P}^*,\mathcal{A})$.*

It will be useful to have the following simple lemma for comparing $\mathsf{Adv}^{\mathsf{predext}}$ with different choices of predicate that are related by logical operators. It can be derived from basic probability calculations.

Lemma 6. *Let PS be a proof system, E be an emulator, Π_1 and Π_2 be predicates, P^* be a cheating prover, and $\mathcal{A}$ be an adversary. Then,*

$$\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_1\vee\Pi_2}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_1\wedge\Pi_2}(\mathsf{P}^*,\mathcal{A}) = \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_1}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_2}(\mathsf{P}^*,\mathcal{A}).$$

[10] The existence a $(\Pi^{\boldsymbol{n}}_{\mathsf{val}} \wedge \Pi^{\boldsymbol{n}}_{\mathsf{nocol}}, \Pi_{\mathsf{wit}} \vee \Pi^*)$-extractor is a natural generalization of special soundness.

and

$$\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_1}(\mathsf{P}^*,\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_1\vee\Pi_2}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\neg\Pi_2}(\mathsf{P}^*,\mathcal{A}).$$

Proof (of Theorem 2). Applying Lemmas 4 and 5, and observing how E is constructed give us the first, third, and fourth claim. For the other claims we need to consider the adversary $\mathcal{B}_{\mathsf{E}}$. Note that it runs E just it would be run in $\mathsf{H}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi^*}(\mathsf{P}^*,\mathcal{A})$, so the distribution over (π,aux) is identical in $\mathsf{H}^{\mathsf{pred}}_{\mathsf{S},\Pi}(\mathcal{B}_{\mathsf{E}})$ as in that game. Furthermore, recall that Π^* is witness-independent, so it ignores its second input. It follows that,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\neg\Pi^*}(\mathsf{P}^*,\mathcal{A}) &= \mathsf{Pr}[\mathsf{V}_\pi(u,tr) \wedge \neg(\neg\Pi^*(\pi,u,\mathrm{aux})) \text{ in } \mathsf{H}^{\mathsf{predext}}] \\ &\leqslant \mathsf{Pr}[\Pi^*(\pi,u,\mathrm{aux}) \text{ in } \mathsf{H}^{\mathsf{predext}}] \\ &= \mathsf{Pr}[\Pi^*(\pi,\varepsilon,\mathrm{aux}) \text{ in } \mathsf{H}^{\mathsf{pred}}] = \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{S},\Pi}(\mathcal{B}_{\mathsf{E}}).\end{aligned}$$

The claimed runtime of $\mathcal{B}$ is clear from its pseudocode (noting that the view of E is distributed identically to its view in $\mathsf{H}^{\mathsf{predext}}$ so its expected number of NEXT queries is unchanged).

For the second claim, we perform the calculations

$$\begin{aligned}\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_{\mathsf{wit}}}(\mathsf{P}^*,\mathcal{A}) &\leqslant \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_{\mathsf{wit}}\vee\Pi^*}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\neg\Pi^*}(\mathsf{P}^*,\mathcal{A}) \\ &= \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi^n_{\mathsf{val}}\wedge\Pi^n_{\mathsf{nocol}}}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PS},\Pi^*}(\mathcal{B}) \\ &= \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi^n_{\mathsf{val}}}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi^n_{\mathsf{nocol}}}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PS},\Pi^*}(\mathcal{B}) \\ &\leqslant 5\mu N/\sqrt{2p} + \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PS},\Pi^*}(\mathcal{B}).\end{aligned}$$

This sequence of calculation uses (in order) Lemma 6, Lemma 5 and the bound we just derived, Lemma 6 (again), and Lemma 4.

4.3 Concrete Bounds on Soundness

Now we discuss how the forking lemma we just derived can be used to provide concrete bounds on soundness. First we make the generic observation that witness-extended emulation implies soundness. Then we discuss how we can use these results together with our expected-time generic group model bound on discrete log security to give concrete bounds on the soundness of various proof systems based on discrete log security, in particular giving the first concrete bound on the soundness of the Bulletproofs proof system for arithmetic circuits.

WITNESS-EXTENDED EMULATION IMPLIES SOUNDNESS. The following theorem observes that finding a witness for u cannot be much more difficult that convincing a verifier u if an efficient witness-extended extractor exists.

Theorem 3. *Let* $\mathsf{PS} = (\mathsf{S},\mathsf{R},\mathsf{P},\mathsf{V},\mu)$ *be a proof system,* G *be a statement generator,* E *be an emulator, and* P^* *be a cheating prover. Define* $\mathcal{A}$ *and* $\mathcal{B}$ *as shown in Fig. 12. Then,*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*) \leqslant \mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}}(\mathsf{P}^*,\mathcal{A}) + \mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_{\mathsf{wit}}}(\mathsf{P}^*,\mathcal{A}).$$

Adversary $\mathcal{A}(\pi)$	Adversary $\mathcal{B}(\pi, u, s)$
$(u, s) \leftarrow\!\!\$\ G(\pi)$	$i \leftarrow 0$
Return $(u, s, (\pi, u))$	$(\cdot, w) \leftarrow\!\!\$\ \mathsf{E}^{\textsc{Next},\textsc{Rew}}(\pi, u)$
$\mathcal{A}(tr, \sigma_{\mathcal{A}})$	Return w
$(\pi, u) \leftarrow \sigma_{\mathcal{A}}$	$\textsc{Next},\textsc{Rew}$
$b' \leftarrow \mathsf{V}_{\pi}(u, tr)$	//Defined as in Fig. 8
Return b'	

Fig. 12. Adversaries used in Theorem 3.

The runtime of that $\mathcal{A}$ is roughly that of G plus that of V. The runtime of $\mathcal{B}$ is roughly that of E when given oracle access to P^* and V interacting.

Proof (Sketch). The use of V in $\mathcal{A}$ ensures that the probability E outputs an accepting transcript must be roughly the same as the probability that P^* convinces V to accept. The difference between these probabilities is bounded by $\mathsf{Adv}^{\mathsf{emu}}_{\mathsf{PS},\mathsf{E}}(\mathsf{P}^*, \mathcal{A})$. Then the Π_{wit} security of E ensures that the probability it outputs a valid witness cannot be much less than the probability it outputs an accepting transcript. The difference between these probabilities is bounded by $\mathsf{Adv}^{\mathsf{predext}}_{\mathsf{PS},\mathsf{E},\Pi_{\mathsf{wit}}}(\mathsf{P}^*, \mathcal{A})$. Adversary $\mathcal{B}$ just runs E to obtain a witness, so $\mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B})$ is the probability that E would output a valid witness.

Discrete log proof systems. A number of the proof systems in [6,7,24] were shown to have a $(\Pi^{\boldsymbol{n}}_{\mathsf{val}} \wedge \Pi^{\boldsymbol{n}}_{\mathsf{nocol}}, \Pi_{\mathsf{wit}} \vee \Pi^{\mathbb{G},n}_{\mathsf{dl}})$-extractor X. For any such proof system PS, Theorem 3 and Theorem 2 bound the soundness of PS by the discrete log relation security of $\mathbb{G}$ against an expected-time adversary $\mathcal{B}_{\mathsf{E}^{\dagger}[\mathsf{T},\mathsf{X}]}$. Moreover, we can then apply Lemma 3 to tightly bound this adversary's advantage by the advantage of an expected-time adversary against normal discrete log security. *We know* how to bound the advantage of such an adversary in the generic group model from Sect. 3.3.

So to obtain a bound on the soundness of these proof systems in the generic group model we can just apply these results to the proof system. To obtain our final concrete security bound in the generic group model we need only to read the existing analysis of the proof system and extract the following parameters,

- p: the size of the set V draws the integer component of its challenges from
- $|\mathbb{G}|$: the size of the group used
- $N = \prod_{i=1}^{\mu} \boldsymbol{n}_i$: the size of the tree that X requires
- $n \geqslant 1$: the number of group elements in the discrete log relation instance
- q_{V}: the number of multi-exponentiations V performs[11]
- q_{X}: the number of multi-exponentiations that X performs

We say such a proof system $\mathsf{PS} = (\mathsf{S}, \mathsf{R}, \mathsf{P}, \mathsf{V}, \mu)$ and extractor X *have parameters* $(p, |\mathbb{G}|, N, n, q_{\mathsf{V}}, q_{\mathsf{X}})$. We obtain the following theorem for such a system, bounding its soundness in the generic group model.

[11] Note that the size of these multi-exponentiations does not matter.

Theorem 4. *Let* $\mathsf{PS} = (\mathsf{S}, \mathsf{R}, \mathsf{P}, \mathsf{V}, \mu)$ *be a proof system and* X *be an extractor that has parameters* $(p, |\mathbb{G}|, N, n, q_{\mathsf{V}}, q_{\mathsf{X}})$. *Let* G *be a statement generator performing at most* q_G *multi-exponentiations and* P^* *be a cheating prover that performs at most* q_{P^*} *multi-exponentiations each time it is run. Define* $\mathcal{B}$ *as shown in Fig. 12. Then in the generic group model we have,*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*) \leqslant \mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B}) + 5\sqrt{\frac{6 \cdot Q_{\mathcal{C}}^2}{|\mathbb{G}|}} + \frac{2}{|\mathbb{G}|} + \frac{5\mu N}{\sqrt{2p}}$$

where $Q_{\mathcal{C}} = q_G + (\mu N + 1)q_{\mathsf{P}^*} + q_{\mathsf{X}} + Nq_{\mathsf{V}} + n + 1$. *The runtime of* $\mathcal{B}$ *is roughly that of* $\mathsf{E}^{\dagger}[\mathsf{T}, \mathsf{X}]$ *when given oracle access to* P^* *and* V *interacting.*

Proof. The result follows by applying Theorem 3, Theorem 2, Lemma 3, and the generic group model bound from Sect. 3.3 as discussed above. □

Concrete security of bulletproofs. Finally, we can use the above to obtain a concrete security bound on the soundness of the Bulletproofs proof system for arithmetic circuits of Bünz et al. [7][12]. To do so we only need to figure out the parameters discussed above. Suppose the proof system is being used for an arithmetic circuit with M multiplication gates. Using techniques of BCCGP [6] this is represented by a size M Hadamard product and $L \leqslant 2M$ linear constraints. Then per Bünz et al. the proof system has the following parameters:

- $p = (|\mathbb{G}| - 1)/2$[13]
- $|\mathbb{G}|$ is the size of group $\mathbb{G}$ in which discrete logs are assumed to be hard
- $N = 7(L + 1)M^3$
- $n = 2M + 2$
- $q_{\mathsf{V}} = 3M + \log_2(M) + 4$
- $q_{\mathsf{X}} = 0$

Having proven our discrete log bound in a generic group model allowing multi-exponentiations is helpful here; it makes our bound not depend on the size of V's multi-exponentiations.

Corollary 1. *Let* PS *be the Bulletproofs proof system for arithmetic circuits define in Sect. 5.2 of [7] using a group of size* $|\mathbb{G}|$. *Let* M *denote the number of multiplication gates in the circuit and* $L \leqslant 2M$ *the number of linear constraints. Let* G *be a statement generator performing at most* q_G *multi-exponentiations and* P^* *be a cheating prover that performs at most* q_{P^*} *multi-exponentiations*

[12] In particular, throughout this section we refer to the logarithmic-sized arithmetic circuit protocol described in Section 5.2 of their paper.

[13] As described in [7], the challenges are drawn from $\mathbb{Z}^*_{|\mathbb{G}|}$. For some rounds of the protocol $x, y \in \mathbb{Z}^*_{|\mathbb{G}|}$ would be considered colliding if $x \equiv_{|\mathbb{G}|} \pm y$. We capture this by thinking of coins drawn from $\{+, -\} \times \mathbb{Z}_p$. Then $(+, x)$ represents $x + 1 \in \mathbb{Z}^*_{|\mathbb{G}|}$ and $(-, x)$ represents $-x - 1 \bmod |\mathbb{G}| = |\mathbb{G}| - x - 1 \in \mathbb{Z}^*_{|\mathbb{G}|}$. Hence the collision condition corresponds to equality in the $\mathbb{Z}_p$ component.

each time it is run. Define $\mathcal{B}$ *as shown in Fig. 12. Assume* $|\mathbb{G}| \geqslant 2$*,* $L \geqslant 1$*, and* $M \geqslant 16$*. Then in the generic group model,*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{PS},G}(\mathsf{P}^*) < \mathsf{Adv}^{\mathsf{wit}}_{\mathsf{PS},G}(\mathcal{B}) + \frac{13q_G + 258q_{\mathsf{P}^*} \cdot LM^3 \log_2(M) + 644 \cdot LM^4}{\sqrt{|\mathbb{G}|}}.$$

The runtime of $\mathcal{B}$ *is roughly that of* $\mathsf{E}^\dagger[\mathsf{T}, \mathsf{X}_B]$ *when given oracle access to* P^* *and* V *interacting, where* X_B *is the Bulletproofs extractor.*

We expect q_{P^*} to be the largest of the parameters, so the bound is dominated by the $O\left(q_{\mathsf{P}^*} \cdot LM^3 \log_2(M)/\sqrt{|\mathbb{G}|}\right)$ term.

Proof. The bound was obtained by plugging our parameters (and $\mu = 3 + \log_2(M)$) into Theorem 4, then simplifying the expression using that $|\mathbb{G}| \geqslant 2$, $L \geqslant 1$, and $M \geqslant 16$. The (straightforward) details of this are provided in the full version of this paper [13]. □

Acknowledgments. We thank Ahsrujit Ghoshal for extensive discussions and his involvement in the earlier stages of this work. We also thank Benedikt Bünz for some clarification regarding [7].

This work was partially supported by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award.

References

1. Barak, B., Lindell, Y.: Strict polynomial-time in simulation and extraction. In: 34th ACM STOC, pp. 484–493. ACM Press (2002)
2. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani, S., di Vimercati, (eds.) ACM CCS 2006, pp. 390–399. ACM Press (2006)
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93, pp. 62–73. ACM Press (1993)
4. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
5. Bernstein, D.J.: How to stretch random functions: the security of protected counter sums. J. Cryptol. **12**(3), 185–192 (1999)
6. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
7. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press (2018)
8. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_24

9. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_19
10. Goldreich, O.: On expected probabilistic polynomial-time adversaries: a suggestion for restricted definitions and their benefits. J. Cryptol. **23**(1), 1–36 (2010)
11. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_22
12. Hoang, V.T., Tessaro, S.: Key-alternating ciphers and key-length extension: exact bounds and multi-user security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 3–32. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_1
13. Jaeger, J., Tessaro, S.: Expected-time cryptography: generic techniques and applications to concrete soundness. Cryptology ePrint Archive (2020). http://eprint.iacr.org/2020/
14. Katz, J., Lindell, Y.: Handling expected polynomial-time strategies in simulation-based security proofs. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 128–149. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_8
15. Katz, J., Lindell, Y.: Handling expected polynomial-time strategies in simulation-based security proofs. J. Cryptol. **21**(3), 303–349 (2008)
16. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptol. **16**(3), 143–184 (2003)
17. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_8
18. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
19. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_8
20. Patarin, J.: The "coefficients H" technique (invited talk). In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_21
21. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
22. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
23. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004). http://eprint.iacr.org/2004/332
24. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy, pp. 926–943. IEEE Computer Society Press (2018)