

Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

NON-MALLEABLE RANDOMNESS
EXTRACTORS

PhD Dissertation

KONRAD DURNOGA



under supervision of
Professor Jacek Pomykała

June 2013

*Die ganze Zahl schuf der liebe Gott, alles Übrige ist Menschenwerk.
(God made the integers, all else is the work of man.)*

— Leopold Kronecker

DECLARATION

Aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

Warsaw, June 2013

Konrad Durnoga

SUPERVISOR'S DECLARATION

This dissertation is ready to be reviewed.

Warsaw, June 2013

Professor Jacek Pomykała

ABSTRACT

We give an unconditional construction of a non-malleable extractor improving the solution from the recent paper *Privacy Amplification and Non-Malleable Extractors via Character Sums* by Dodis *et al.* (FOCS'11). There, the authors provide the first explicit example of a non-malleable extractor – a cryptographic primitive that significantly strengthens the notion of a classical randomness extractor. In order to make the extractor robust, so that it runs in polynomial time and outputs a linear number of bits, they rely on a certain conjecture on the least prime in a residue class. In this dissertation we present a modification of their construction that allows to remove that dependency and address an issue we identified in the original development. Namely, it required an additional assumption about feasibility of finding a primitive element in a finite field. As an auxiliary result, which can be of independent interest, we show an efficiently computable bijection between any order M subgroup of the multiplicative group of a finite field and a set of integers modulo M with the provision that M is a smooth number. Also, we provide a version of the baby-step giant-step method for solving multiple instances of the discrete logarithm problem in the multiplicative group of a prime field. It performs better than the generic algorithm when run on a machine without constant-time access to each memory cell, e.g., on a classical Turing machine.

Keywords: randomness extractor, non-malleable extractor, discrete logarithm, baby-steps giant-steps, derandomization, algorithm with external memory.

ACM Classification: F.1.1, F.1.2, F.2.1

STRESZCZENIE

Rozprawa poświęcona jest analizie *ekstraktorów losowości*, czyli deterministycznych funkcji przekształcających niedoskonałe źródła losowości na takie, które są w statystycznym sensie bliskie rozkładom jednostajnym. Główny rezultat dysertacji stanowi bezwarunkowa i efektywna konstrukcja ekstraktora pewnego szczególnego typu, zwanego *ekstraktorem niekowalnym*. Jest to poprawienie wyniku z opublikowanej niedawno pracy *Privacy Amplification and Non-Malleable Extractors via Character Sums* autorstwa Dodisa *i in.* (FOCS'11). Podana tam konstrukcja stanowiła pierwszy jawny przykład ekstraktora niekowalnego, choć był to rezultat warunkowy, odwołujący się do pewnej hipotezy dotyczącej liczb pierwszych w postępach arytmetycznych. W rozprawie przedstawiona jest modyfikacja rozwiązania Dodisa *i in.*, która pozwala na usunięcie tego dodatkowego założenia. Jednocześnie wskazana w dysertacji i występująca w oryginalnym rozumowaniu luka, związana z problemem wydajnego znajdowania generatora grupy moltiplicatywnej w ciele skończonym, nie przenosi się na proponowaną w rozprawie konstrukcję.

Słowa kluczowe: ekstraktor losowości, ekstraktor niekowalny, logarytm dyskretny, algorytm małych-wielkich kroków, derandomizacja, algorytm z pamięcią zewnętrzną

Klasyfikacja wg ACM: F.1.1, F.1.2, F.2.1

ACKNOWLEDGMENTS

I would like to thank Professor Stefan Dziembowski for introducing me to the topic of randomness extractors. My thanks also go to members of the Cryptology and Data Security Group from University of Warsaw, in particular: Dr. Tomasz Kazana, Dr. Maciej Obremski, and Michał Zając.

I appreciate guidance of my advisor, Professor Jacek Pomykała, thanking for his help and understanding during my PhD studies. I express my deep gratitude to Dr. Bartosz Żrałek, a coauthor of the manuscript (Durnoga & Żrałek 2013) this thesis is based upon.

I am obliged to Professor Carl Pomerance for his hints on the proof of [Theorem 3.8](#). I also offer my thanks to Professor Joachim von zur Gathen for reading the preprint (Durnoga & Żrałek 2013) and pointing some references on generating safe primes and finding elements of large orders in finite fields.

I acknowledge the financial support provided to me by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no CNTM-207908.

Last but not least, I am indebted to my parents for their continuous support from my very beginning. This thesis would not have been possible without you.

CONTENTS

1 INTRODUCTION	1
2 PRELIMINARIES	5
3 NON-MALLEABLE EXTRACTOR	9
4 ONLINE PSEUDORANDOM GENERATOR	17
5 NON-MALLEABLE EXTRACTOR WITHOUT THE ERH	25
6 EFFECTIVE BIJECTION	29
7 NON-MALLEABLE EXTRACTOR WITH LONG OUTPUT	37
8 COMPUTING DISCRETE LOGARITHMS WITH GAUSSIAN PERIODS	43
BIBLIOGRAPHY	49

INTRODUCTION

One of the pivotal problems in practical computer science (suffice it to mention such areas like scientific numerical computations, probabilistic algorithms, or cryptographic purposes) is providing users with “good” sources of randomness. Here, the word *good* refers to statistical properties of a source which is supposed to produce truly random bits, or at least to be close to uniform. Such requirements can be accomplished using physical devices yet with a caveat that specialized hardware can be relatively expensive and not as efficient as needed in practice. Thus there emerges a need to construct effective *extractors* – deterministic algorithms capable of transforming a (cheap) source of weak randomness into a one being close to uniform distribution. Employing an extractor allows us to amplify the so called min-entropy that measures quality of randomness of a source.

The history of this topic goes back to the '50s and credit should be given to von Neumann for his pioneer work on unbiasing a biased coin. The term *extractor* was devised by [Nisan & Zuckerman \(1993\)](#) but the foundations of this field had been laid earlier by [Chor & Goldreich \(1988\)](#), [Cohen & Wigderson \(1989\)](#), and [Zuckerman \(1990\)](#). One of the first explicit constructions of extractors stem from works of [Chor & Goldreich](#), or of [Impagliazzo *et al.* \(1989\)](#) on hash function families. Since the late '80s the problem has drawn attention of many researchers which resulted in numerous papers (see, e.g., a survey by [Shaltiel \(2002\)](#) for an extensive list of references), one of the most notable being [Trevisan's](#) development built on pseudorandom generators. A major breakthrough was due to [Bourgain \(2005\)](#) who challenged the problem of breaking the famous $1/2$ -rate barrier.

An important contribution of [Dodis *et al.* \(2011\)](#) is showing that the Chor-Goldreich extractor enjoys a non-malleability property, which is apparently very strong. The term *non-malleable extractor* was coined by [Dodis & Wichs \(2009\)](#) who, using this primitive, gave a protocol for privacy amplification that remains secure in presence of an active adversary. One considerable deficiency of the solution by [Dodis *et al.*](#) is that it relies on certain long-standing conjecture on the least prime in arithmetic progression. In this dissertation, we slightly modify their construction and get an unconditional result. Also, we identify a pitfall in their development which, in essence, is that the authors assume a primitive element of a finite field is given. However, they do not elaborate on the exact way of how this element can be chosen (this was overlooked by the authors as confirmed in a private communication), and, in fact, the question whether it is possible to generate such an element in, say, $\mathbb{Z}/p\mathbb{Z}$ for large p in polynomial time in $\log p$ remains an open problem when the factorization of $p - 1$ is unknown. We address this particular issue of the construction in subsequent chapters. We note that a shortcoming of a similar nature is present in a work by [Li \(2012b\)](#) which also covers non-malleable extractors.

There are several recent papers, e.g., by [Cohen *et al.* \(2012\)](#), [Li \(2012b\)](#), and [Li \(2012a\)](#), discussing the non-malleability property of extractors and related topics. They offer solutions superior, in terms of min-entropy rate or error bound, to the construction of [Dodis & Wichs](#) which, chronologically, is the earliest example of a non-malleable extractor. These results were obtained using different means that supposedly have little to do with the number-theoretic approach of [Chor & Goldreich](#).

As a by-product of our considerations on non-malleable extractors we come up with a method for calculating certain bijective mapping between a subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$ of large order M and $\mathbb{Z}/M\mathbb{Z}$. A running time of our algorithm is roughly proportional to the largest prime factor of M . The problem of finding such an algorithm which is polynomial in $\log p$ can be interesting on its own. This would have a remarkable impact in practical cryptogra-

phy. For instance, it is sensible to ask how to extract a random key from a random group element, e.g., in the Diffie-Hellman protocol. [Chevalier *et al.* \(2009\)](#) show a simple shrinking function that trims bit representation of a group element to its least significant bits and produces satisfactory, if not optimal, results. Yet, computing an exact bijection is known to be feasible only for certain subgroups of $(\mathbb{Z}/p\mathbb{Z})^*$, e.g., the subgroup of quadratic residues mod p if $(p - 1)/2$ is a Sophie Germain prime.

The bottom line of the Chor-Goldreich extractor is computing a discrete logarithm. The discrete logarithm problem (DLP) is ubiquitous in cryptography, but here, unlike in other cryptographic applications, we are interested in cases where this problem is actually easy to solve. This is done by the standard algorithm of [Pohlig & Hellman \(1978\)](#) together with Shanks' baby-step giant-step (BSGS) method. We invoke it in several places and finally tailor the generic BSGS algorithm to solve multiple instances of the DLP in the multiplicative group of a prime field. The modification does not offer any substantial improvements in the common computation model, i.e., the RAM, where memory can be accessed in constant time. However, in certain subgroups of $(\mathbb{Z}/p\mathbb{Z})^*$ it outperforms the BSGS method when run on a Turing machine. As argued by [Diem \(2012\)](#), the motivation behind switching to the Turing model is that the algorithm can possibly operate on large amounts of data which cannot be fit into computer's main memory at once. It thus has to reside on some slow external storage that is abstracted with a machine's tape. Historically, the very same argument led to developing algorithms for external sorting, such as the mergesort. [Diem](#) reviews some nowadays standard number-theoretic algorithms, including the BSGS method, implemented on a multitape Turing machine. Where our solution departs from the typical approach is in the fact that it does not involve sorting large arrays. It is thus aimed at machines with a single work-tape, where sorting is not a viable option.

The algorithms we propose in the dissertation are aimed at processing multiple inputs. For instance, we present a method to cal-

culate an extractor-backing function for many arguments. In some sense this algorithm is capable of “streaming” nearly random values. Its real-life counterparts are (pseudo-)random generators in computer systems, yet with the difference that our program is not initialized with a seed, but instead it has to be fed with appropriate imperfect randomness repeatedly. An important feature our algorithms enjoy is that they process their inputs sequentially. That is, a portion of input is read, processed, and an answer is returned before moving on to the next portion. Algorithms of this kind, called *online algorithms*, are fundamental in interactive computations, where the whole input is unavailable at start. Online algorithms have been extensively studied, see, e.g., the survey by [Fiat & Woeginger \(1998\)](#), and the interest they receive is motivated by their applicability to operating systems and networks.

ORGANIZATION OF THE THESIS. Some basic definitions, we refer to later on, are listed in [Chapter 2](#). For completeness of exposition, in [Chapter 3](#) we recall the construction of a non-malleable extractor given by [Dodis *et al.*](#) and focus on its computational aspect. In [Chapter 4](#), we propose a method to evaluate extractor’s underlying function for multiple arguments without knowing a primitive element in a finite field. A modified version of the extractor, which performs equally well as the original one but does not involve a primitive element, is presented in [Chapter 5](#). Next, in [Chapter 6](#), we show bijective mappings to transform a group element outputted by our extractor to an integer modulo some parameter. [Chapter 7](#) demonstrates that trimming a result of our extractor to least significant bits produces nearly uniformly random values. By this last fact we obtain an unconditional construction of a non-malleable extractor with long output. [Chapter 8](#) contains an algorithm for finding discrete logarithms in $\text{GF}(p)$ that surpasses the usual BSGS method on a classical Turing machine when there are multiple instances of the DLP to solve.

PRELIMINARIES

Throughout the dissertation p and q always denote prime numbers. We use the letter g for a primitive element of a prime field $\mathbb{Z}/p\mathbb{Z}$, i.e., g is a generator of the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$. We let $M > 1$ be a divisor of $p - 1$, possibly subject to some additional restrictions stated separately in due course. For a prime $q \mid M$, we use q^{α_q} to denote the largest power of q dividing M ; in short $q^{\alpha_q} \parallel M$. We write $P^+(M)$ for the largest prime divisor of M . The cardinality of a set \mathcal{Z} is denoted by $|\mathcal{Z}|$. The same notation is used to write the order of a finite group.

For fixed p and M , a subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$ that attracts our particular interest in the remainder of the thesis is defined by

$$G := \{a^{(p-1)/M} \mid a \in (\mathbb{Z}/p\mathbb{Z})^*\}.$$

Closely related to G are its Sylow q -subgroups

$$G_q := \{a^{M/q^{\alpha_q}} \mid a \in G\}$$

for each $q^{\alpha_q} \parallel M$.

COMPUTING DISCRETE LOGARITHMS

We apply the well-known generic baby-step giant-step method by [Shanks \(1971\)](#) to compute discrete logarithms in subgroups of $(\mathbb{Z}/p\mathbb{Z})^*$. Its running time for groups of order q is $O(\frac{q}{s} \text{poly}(\log p))$ if it is allowed $O(s \text{poly}(\log p))$ bits of memory, where the parameter $s \leq q^{1/2}$ can be chosen arbitrarily. Often, we use the BSGS method in conjunction with the Pohlig-Hellman (PH) algorithm for order M subgroups. Then, for any $s \leq \sqrt{P^+(M)}$ the time complexity becomes $O(P^+(M) \cdot s^{-1} \text{poly}(\log p))$ with the same space bound as in the generic BSGS.

Both algorithms are classic, yet there are some more recent developments, e.g., the ones by [Terr \(2000\)](#) or [Stein & Teske \(2005\)](#), that offer slight advantages over the original BSGS method in certain settings. Our construction from [Chapter 8](#) is also in this vein.

All the running costs we consider count the number of bit operations and already include costs of group arithmetic. We use the standard primitives for performing integer multiplication (e.g. the Schönhage–Strassen algorithm) or exponentiation, yet their detailed contribution to the overall time cost is hidden in a $\text{poly}(\log p)$ factor. This factor could be made explicit but it would obscure the general exposition.

MIN-ENTROPY AND PROBABILITY DISTRIBUTIONS

The established way to define a randomness extractor, including a non-malleable one which we introduce next, relies on the well-known notion of min-entropy that measures randomness of a distribution. For a discrete random variable X , the *min-entropy* of X is defined as

$$H_\infty(X) := \min_x \log_2 \frac{1}{\Pr(X = x)},$$

where the minimum is taken over all x from the support of X . In this context, it is a common practice to use the two phrases: a *random variable* and a *probability distribution* interchangeably with a slight abuse of notation. To quantify how similar two variables X and X' , both distributed over the same finite set \mathcal{X} , are, we use a *statistical distance* $\Delta(X, X')$ specified by

$$\begin{aligned} \Delta(X, X') &:= \max_{\mathcal{S} \subseteq \mathcal{X}} |\Pr(X \in \mathcal{S}) - \Pr(X' \in \mathcal{S})| \\ &= \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr(X = x) - \Pr(X' = x)|. \end{aligned}$$

We write $X \approx_\epsilon X'$ to indicate that $\Delta(X, X') \leq \epsilon$ for some $\epsilon \geq 0$. Typically, ϵ is required to be *negligible* in some security parameter,

say n . That is, ϵ as a function of n should vanish faster than the inverse of any polynomial of n .

$U_{\mathcal{Z}}$ stands for a uniform distribution over some finite set \mathcal{Z} .

FOURIER ANALYSIS AND NON-UNIFORM XOR LEMMA

Fourier analysis is a paramount tool in the theory of randomness extractors (see, e.g., the expository work by Rao (2007)). Although we do not use these techniques ourselves in this thesis, we list some basic notions to state results of other authors in subsequent chapters. The notation we use is standard but we adopt the normalization from the work by Dodis *et al.* which differs from the one present in literature.

For two functions $f, f': \mathcal{X} \rightarrow \mathbb{C}$ over some finite set \mathcal{X} their inner product $\langle f, f' \rangle$ is defined by $\langle f, f' \rangle := \sum_{x \in \mathcal{X}} f(x) \overline{f'(x)}$. The ℓ_1 -norm of f is just $\|f\|_{\ell_1} = \sum_{x \in \mathcal{X}} |f(x)|$. A probability distribution X over \mathcal{X} can also be viewed as a real-valued function assigning the probability $\Pr(X = x)$ to each $x \in \mathcal{X}$.

A *character* of a finite abelian group H is any homomorphism $H \rightarrow \mathbb{C}^*$. A character is called *non-trivial* if it is not identically equal to 1. All the characters of H form a finite abelian group, called the dual group of H , with pointwise multiplication as the group operation. The Fourier transform \widehat{f} of a function $f: H \rightarrow \mathbb{C}$ is a function acting on the dual group of H . For every character ϕ of H the Fourier coefficient $\widehat{f}(\phi)$ is defined as $\langle f, \phi \rangle$.

Vazirani's XOR lemma is, by all means, the main link that connects random variables, statistical distances, and Fourier analysis and it appears to be a folklore. For non-malleable extractors a more general version of it, the non-uniform XOR lemma recently devised and proven by Dodis *et al.*, is needed. We formulate it below.

LEMMA 2.1 (Dodis *et al.*, Lemma 3.8). *Let H be a finite abelian group. Suppose that for a pair of random variables Z and Z' distributed over*

H , and all characters ϕ and ϕ' on H with ϕ' non-trivial, we have the following bound on the expectation of $\phi(Z) \cdot \phi'(Z')$:

$$(2.2) \quad |\mathbb{E}_{(Z, Z')}[\phi(Z)\phi'(Z')]| \leq \alpha.$$

Then, $\Delta((Z, Z'), (\mathcal{U}_H, Z')) \leq \frac{1}{2}\alpha \cdot |H|$ for a uniformly distributed variable \mathcal{U}_H which is independent of Z' .

We invoke [Lemma 2.1](#) in the next chapter when discussing non-malleability of the Chor-Goldreich extractor.

NON-MALLEABLE EXTRACTOR

In this chapter, we recall the construction of a discrete logarithm-based randomness extractor. It was proposed in a seminal work by [Chor & Goldreich \(1988\)](#). Below, we follow the exposition given by [Dodis *et al.* \(2011\)](#) and point out some significant differences between these two approaches. We comment on feasibility of evaluating the extractor, in particular, on how its key parameters can be chosen effectively.

DEFINITION

A function $\text{Ext}: \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$ is a (k, ϵ) -*non-malleable extractor* if for any pair of independent random variables X and Y , distributed over \mathcal{X} and \mathcal{Y} , respectively, and every function $A: \mathcal{Y} \rightarrow \mathcal{Y}$ satisfying $A(y) \neq y$ for each $y \in \mathcal{Y}$, we have:

$$(3.1) \quad (\text{Ext}(X, Y), \text{Ext}(X, A(Y)), Y) \approx_{\epsilon} (\mathcal{U}_{\mathcal{Z}}, \text{Ext}(X, A(Y)), Y),$$

provided that $H_{\infty}(X) \geq k$, and Y is uniform on \mathcal{Y} . Intuitively, this definition states that if x is drawn from a sufficiently random distribution, then no adversary can distinguish $\text{Ext}(x, y)$ from a random value, even if he is given a random seed y and $\text{Ext}(x, A(y))$ for a function A the adversary could choose beforehand. We note that this notion offers a significant strengthening of a randomness extractor, where the condition (3.1) is substituted with $\text{Ext}(X, Y) \approx_{\epsilon} \mathcal{U}_{\mathcal{Z}}$, and a strong extractor, where we can use

$$(\text{Ext}(X, Y), Y) \approx_{\epsilon} (\mathcal{U}_{\mathcal{Z}}, Y)$$

in place of (3.1). [Dodis & Wichs \(2009\)](#) came up with a probabilistic argument showing that non-malleable extractors exist under some

plausible assumptions on parameters, yet no explicit construction had been known until the work by [Dodis *et al.* \(2011\)](#).

CONSTRUCTION

Now, suppose that we are given a prime p , a generator g of $(\mathbb{Z}/p\mathbb{Z})^*$, and a divisor $M > 1$ of $p - 1$. The crux of the Chor-Goldreich extractor is the following function $f_g: (\mathbb{Z}/p\mathbb{Z})^* \rightarrow \mathbb{Z}/M\mathbb{Z}$:

$$(3.2) \quad f_g(a) := \log_g a \pmod{M}$$

extended to $\mathbb{Z}/p\mathbb{Z}$ by letting, e.g., $f_g(0) = 0$. [Theorem 3.3](#) declares that one can build a non-malleable extractor based on f_g .

THEOREM 3.3 ([Dodis *et al.*](#), Theorem 4.1). *For a fixed prime p , a divisor $M \mid p - 1$, and a primitive element g , define $\text{Ext}: \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{Z}/M\mathbb{Z}$ by*

$$(3.4) \quad \text{Ext}(x, y) := f_g(x + y) .$$

Then, for any k , the above function Ext is a (k, ϵ) -non-malleable extractor with $\epsilon = 2Mp^{1/4}2^{-k/2}$.

Note that the above theorem can be non-trivial only for source entropy rate greater than $1/2$, i.e., for $k > (1/2 + \delta) \log_2 p$ for some constant $\delta > 0$.

To prove [Theorem 3.3](#) [Dodis *et al.*](#) use the non-uniform XOR lemma ([Lemma 2.1](#)). An ingenious observation is that for any character ϕ of $\mathbb{Z}/M\mathbb{Z}$ the composition $\phi \circ f_g$ corresponds to a Dirichlet (multiplicative) character of $(\mathbb{Z}/p\mathbb{Z})^*$. By making the most of Weil's estimate on character sums over finite fields (see, e.g., the book by [Schmidt \(1976\)](#)) it is possible to handle the expectation on the left-hand side of [\(2.2\)](#).

Prior to this result it had been known that [\(3.4\)](#) defined a randomness extractor, which followed from the initial paper of [Chor & Goldreich](#). Later, [Dodis & Oliveira \(2003\)](#) established that Chor-Goldreich extractor was strong. In fact, the main result [Dodis *et al.*](#)

achieve is slightly more general than [Theorem 3.3](#) in that it holds for any finite field, not necessarily a prime one.

PARAMETERS

A subject of our foremost concern in this thesis is feasibility of computing Ext , or, equivalently, f_g . A variant of the PH algorithm, where discrete logarithms are calculated modulo each $q^{\alpha_q} \parallel M$, allows for evaluating f_g in $O(P^+(M) \text{poly}(\log p))$ steps. This is polynomial time when M is sufficiently smooth. A problem that arises here is how to generate suitable p and M with $M \mid p - 1$. One can decide on the following approach. In order to have an extractor outputting m bits, we pick $M = 2^m$. Given the values of k and ϵ we expect to achieve, we take an integer $n > m$ to be an approximation of $\log_2 p$ backtracked from the formula for ϵ of [Theorem 3.3](#). Let $N = 2^n$. We choose p to be the *smallest* prime appearing in the residue class $1 \pmod{N}$. Each candidate can be tested for primality in deterministic polynomial time, e.g., using the algorithm by [Agrawal et al. \(2002\)](#) or [Lenstra \(2002\)](#), and the searching procedure is effective under a widely-believed conjecture (cf. [Granville & Pomerance \(1990\)](#)) on primes in arithmetic progressions.

CONJECTURE 3.5. *For any a and d with $\gcd(a, d) = 1$ the least prime $p(a, d)$ in the arithmetic progression $a \pmod{d}$ is $O(\varphi(d) \log^2 d)$.*

An argument of this kind is also given by [Dodis et al.](#), but actually their work contains an oversight in that it suggests looking at the progression $\text{mod } M$ instead of $\text{mod } N$. This, however, does not guarantee a proper magnitude of a prime found in the process. For instance, it could happen that $M + 1$ is prime, yet $p = M + 1$ would yield non-negligible ϵ in [Theorem 3.3](#), regardless of the min-entropy bound k . Also, for this reason one cannot apply, as suggested by [Dodis et al.](#), any of unconditional results on Linnik's constant (e.g. $p(a, d) = O(d^5)$) recently established by [Xy-](#)

louris (2011)) in the case where M is small, say $M = O(\text{poly}(k)) = O(\text{poly}(\log N))$. Despite this fact, it is possible to generate p for such M without relying on Conjecture 3.5, which comes at the price of introducing nondeterminism. Namely, for small M and N as above, we sample l uniformly at random such that $l \equiv 1 \pmod{M}$ and $N < l \leq 2N$. If l is prime then we pick $p := l$, repeating the procedure otherwise. The expected number of trials is $O(\log N)$ which follows from the below corollary of the Siegel-Walfisz theorem (cf. Iwaniec & Kowalski (2004)).

THEOREM 3.6 (Siegel-Walfisz). *Let $\pi(x; a, d)$ be the function counting the number of primes up to x which are congruent to $a \pmod{d}$, where $\gcd(a, d) = 1$. Then, for any constant $C > 0$, it holds that*

$$(3.7) \quad \pi(x; a, d) = \frac{\pi(x)}{\varphi(d)} + O_C(x \log^{-C} x).$$

An inherent downside of (3.7) is that the implied big-O constant, which depends on C , is not effectively computable. The error term in (3.7) limits applicability of the above theorem to small d , i.e., $d = (\log x)^{O(1)}$. We note, however, that the asymptotic relation $\pi(x; a, d) \sim \pi(x)/\varphi(d)$ is purportedly true for much broader range of moduli d . Namely, under the Extended Riemann Hypothesis (ERH) the error term in (3.7) can be reduced to $O(x^{\frac{1}{2}} \log^{-C} x)$ for any constant $C > 0$.

Another approach to generating p and M that only slightly differs from the previous ones would be finding a random prime p of $\log N$ bits first, factoring $\text{poly}(\log N)$ -smooth part of $p - 1$, and letting M to be a divisor of this part. To estimate the expected number of p that need to be tested until M of preselected magnitude is found one can turn to density results for smooth shifted primes. If we let $\pi(x, y)$ count primes $p \leq x$ such that $p - 1$ is y -smooth and write $u = \log x / \log y$, then it is believed that $\pi(x, y) \sim \rho(u)\pi(x)$ holds (see, e.g., the article by Pomerance & Sharlinski (2002) for this conjecture and the current state of the art)

for a wide range of u , where $\rho(u) = u^{-u+o(u)}$ is the Dickman-de Bruijn function. Unfortunately, even if this relation was true for large u (like in the setting: $x = N$, $y = O(\text{poly}(\log N))$, and $u = \Omega(\log N / \log \log N)$) then the expected searching time for p and M would be $\Omega(\rho(u)^{-1})$, which is unacceptably large (exponential in $\log N$ in our example). Readjusting some parameters brings this complexity back to the polynomial level. Also, we only need $p - 1$ to have a large smooth divisor but $p - 1$ itself does not have to be smooth. [Chor & Goldreich](#) take the following path: fix a parameter n , let $N := n^{\sqrt{\log \log n}}$, and search for a prime p such that $N < p \leq N^2$ with $p - 1$ having an n -smooth part M between $p^{3/16}$ and $p^{1/4}$, where the actual exponents $3/16$ and $1/4$ are just an artifact of their proof method which differs significantly from the one used later by [Dodis et al.](#) and, in particular, does not work for small M . The authors then claim that the search procedure is effective as the probability of finding a suitable pair (p, M) is $\Omega(1/\log^2 n)$. This estimate comes from a private communication with Carl Pomerance and is stated without a rigorous proof. It can be inferred indeed from the below theorem.

THEOREM 3.8. *There exist efficiently computable positive constants x_0 and C such that for every $x > x_0$ and $y > \log x$ the number of primes $p \leq x$ with $p - 1$ having a y -smooth divisor exceeding $x^{1/3}$ is at least $(Cu \log \log y)^{-u/3} \cdot x / \log x$, where $u = \log x / \log y$.*

As Carl Pomerance has pointed us, the theorem can be established by modifying the reasoning given by [Konyagin & Pomerance \(1996\)](#) (cf. the proof of Theorem 5.2 there) that relies on the powerful result by [Alford et al. \(1994\)](#). The latter is excerpted as [Theorem 5.2 in Chapter 5](#), where we reuse it to obtain another, yet simple, estimate for primes in arithmetic progressions.

We note that what [Chor & Goldreich](#) require is finding the full factorization of $p - 1$ for each candidate p . This can be done by an algorithm which is subexponential in $\log p$ but polynomial in n . Then, a probabilistic primality test is run in attempt to construct

a certificate using Pratt’s method. As an important by-product it yields a primitive element g in $\mathbb{Z}/p\mathbb{Z}$ if p happens to be prime. Overall, [Chor & Goldreich](#) manage to construct a two-source extractor with the error bound $\epsilon = O(n^{-2\sqrt{\log \log n}})$. This is clearly negligible in n yet it is a bit uncommon to let the key parameters p , M , and ϵ^{-1} be only subexponential in the security parameter.

Getting back to the work by [Dodis *et al.*](#), the last parameter that has to be determined is g , but the authors do not comment on its choice in the paper. In general, the problem of finding a primitive element in a large finite field in deterministic polynomial time remains open. On the other hand, if we have the full prime factorization of $p - 1$ then there is an obvious randomized algorithm to get such g efficiently. Now, if [Conjecture 3.5](#) is assumed then $p - 1$ is $O(\log^2 p)$ -smooth and its factorization can be found with ease. This procedure can be further derandomized under the ERH. Namely, the theorem by [Ankeny \(1952\)](#) asserts that every proper subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$ omits a small element, of magnitude $O(\log^2 p)$, in $(\mathbb{Z}/p\mathbb{Z})^*$. This leads to a deterministic algorithm for finding a generator of the whole group $(\mathbb{Z}/p\mathbb{Z})^*$. The big-O constant in the last bound was made explicit by several authors with the up to date record $\frac{3}{2}$ held by [Wedeniwski \(2001\)](#). In the light of a conditional result by [Shoup \(1990\)](#) there is a primitive element among the first $O(\log^6 p)$ elements of $\mathbb{Z}/p\mathbb{Z}$. Later, also assuming the ERH, [Bach \(1997\)](#) showed how to construct, in deterministic polynomial time, a set of moderate size, i.e., of $o(\log^4 p)$ residues, that contains a primitive element.

Surprisingly, the problem of finding g gets more interesting in the seemingly simpler case when M is small, where [Conjecture 3.5](#) is not needed and the factorization is not known. We observe, however, that even in the general case what we actually need is an element of order being equal to M or some multiple of M in $(\mathbb{Z}/p\mathbb{Z})^*$, but not necessarily a primitive element. That said, if we had an element, say h , with $M \mid \text{ord}(h)$ we could modify our extractor first by sending $x + y$ to $\langle h \rangle$ and then taking base

h logarithm modulo M of the resulting image, i.e., $\text{Ext}'(x, y) := \log_h(x + y)^{(p-1)/\text{ord}(h)} \bmod M$.

The task of generating an element of possibly large order is a bit easier. For instance, [Gao \(1997\)](#) shows how to construct an element of $\text{GF}(p^l)$ with order superpolynomial in l . This result can be useful for the extractor in non-prime fields. Recently, Joachim von zur Gathen has informed us that [Gao's](#) result can be improved to exponential orders but generated elements come from a small extension of $\text{GF}(p^l)$ and not the base field itself ([von zur Gathen & Shparlinski 2001](#)). The aforementioned theorem of [Ankeny](#) implies that an element of order M can be found efficiently under the ERH. Finally, the profound work by [Pila \(1990\)](#) gives an unconditional algorithm for prime M with $(\log p)^{O_M(1)}$ running time in $\mathbb{Z}/p\mathbb{Z}$ but the exponent $O_M(1)$ depends on M . This allows us to cover the case of extractors having output of constant length. However, we are still short of a method that works unconditionally for general M . In [Chapter 5](#) we explore the above idea of mapping $x + y$ to some group of large order, which enables us to get rid of generating g (or h) and computing discrete logarithms.

ONLINE PSEUDORANDOM GENERATOR

Below, we devise an algorithm for computing the function f_g , given by (3.2), where a primitive element g is not known in advance. Arguably, this task may not seem well posed as we cannot viably calculate an unknown function in a deterministic way. What we actually propose is a method for evaluating a certain partial function f for multiple arguments a_i , for say $i = 1, \dots, \ell$, ensuring there always exists g such that $f(a_i) = f_g(a_i)$ for each a_i . The value of g depends on the given sequence of a_i . There may be more than one g yielding f_g which is consistent with f , yet our algorithm offers no feasible method to infer any of them explicitly. A motivation behind such an approach is that f_g gives rise to a non-malleable extractor no matter which g is plugged into it.

AUXILIARY LEMATA

Before we proceed to the algorithm, we prove two auxiliary facts. Everywhere below when we use the term *homomorphism*, possibly an injective or bijective one, we refer to a mapping between a subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$ and an additive group modulo some integer, typically $G \rightarrow \mathbb{Z}/M\mathbb{Z}$ or $G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$, where $q^{\alpha_q} \parallel M$. The following lemma relates f and f_g characterizing these primitive elements g for which both functions are identical.

LEMMA 4.1. *Let $\psi: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ be an isomorphism and let a function $f: (\mathbb{Z}/p\mathbb{Z})^* \rightarrow \mathbb{Z}/M\mathbb{Z}$ be given by*

$$f(a) := \psi(a^{(p-1)/M}).$$

Then, $f = f_g$ for every primitive element g such that

$$g^{(p-1)/M} = \psi^{-1}(1).$$

PROOF. Write g' for the unique element of G satisfying $\psi(g') = 1$. Consider any primitive element g of $\mathbb{Z}/p\mathbb{Z}$ such that $g^{(p-1)/M} = g'$. Take any $a \in (\mathbb{Z}/p\mathbb{Z})^*$ and let $u := \log_g a$. Now, $g'^{u \bmod M} = a^{(p-1)/M}$ and hence

$$f_g(a) = u \bmod M = (u \bmod M)\psi(g') = \psi(a^{(p-1)/M}),$$

as claimed in the lemma. \square

The below lemma is just an application of the Chinese remainder theorem (CRT) to bijective mappings $G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$.

LEMMA 4.2. *Suppose that for each $q^{\alpha_q} \parallel M$ there exists a bijection $\psi_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$. Then, the mapping $\psi: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ defined by means of the CRT by*

$$(4.3) \quad \psi(a) := \psi_q(a^{M/q^{\alpha_q}}) \bmod q^{\alpha_q} \quad \text{for each } q^{\alpha_q} \parallel M$$

is a bijection. Moreover, if each ψ_q is an isomorphism then so is ψ .

PROOF. Since $|G| = M = |\mathbb{Z}/M\mathbb{Z}|$, it suffices to show that ψ , as given by (4.3), is injective. To this end, suppose that $\psi(a) = \psi(b)$ for some $a, b \in G$. In other words, it holds that $\psi_q(a^{M/q^{\alpha_q}}) = \psi_q(b^{M/q^{\alpha_q}})$ for every $q^{\alpha_q} \parallel M$. By the fact that each ψ_q is a one-to-one function, this relation is equivalent to $a^{M/q^{\alpha_q}} = b^{M/q^{\alpha_q}}$. Clearly, $\gcd\{M/q^{\alpha_q} \mid q \text{ is a prime, } q^{\alpha_q} \parallel M\} = 1$, so there exist integers r_q satisfying $\sum' r_q M/q^{\alpha_q} = 1$, where the sum \sum' ranges over all $q^{\alpha_q} \parallel M$. Hence,

$$a = a^{\sum' r_q M/q^{\alpha_q}} = b^{\sum' r_q M/q^{\alpha_q}} = b$$

and ψ is indeed injective.

Additionally, if each ψ_q is a homomorphism, then for $a, b \in G$ we obtain that $\psi(ab) = \psi(a) + \psi(b)$ simply by adding two formulæ (4.3) for $\psi(a)$ and $\psi(b)$, and applying the CRT. \square

ALGORITHM

In the light of the above lemata, our task boils down to constructing an online algorithm that computes some isomorphism $G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$. We outline it as [Algorithm 4.4](#). The idea is to keep track of a generator of a currently known subgroup (generated by all inputs processed previously) of G_q and, each time when the subgroup grows after adding a new input, to extend the constructed monomorphism in a consistent way.

ALGORITHM 1. Online computing of ψ_q .

Input: A sequence $a_1, \dots, a_\ell \in (\mathbb{Z}/p\mathbb{Z})^*$

Output: A sequence v_1, \dots, v_ℓ such that $\psi_q(a_i^{(p-1)/q^{\alpha_q}}) = v_i$ for $i = 1, \dots, \ell$ and some isomorphism $\psi_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$ depending on a_1, \dots, a_ℓ

1. $b_0 \leftarrow 1$
 2. $g_0 \leftarrow b_0 \quad e_0 \leftarrow q^{\alpha_q}/\text{ord}(g_0)$
 3. For $i \leftarrow 1, \dots, \ell$ do [4-14](#)
 4. $b_i \leftarrow a_i^{(p-1)/q^{\alpha_q}}$
 5. If $\text{ord}(b_i) \leq \text{ord}(g_{i-1})$ then
 6. compute $0 \leq u < q^{\alpha_q}$ such that $g_{i-1}^u = b_i$
 7. $v_i \leftarrow u \cdot e_{i-1} \bmod q^{\alpha_q}$
 8. $g_i \leftarrow g_{i-1} \quad e_i \leftarrow e_{i-1}$
 9. Else
 10. compute $0 < u < q^{\alpha_q}$ such that $b_i^u = g_{i-1}$
 11. let $u = q^\beta \cdot w$ where $q \nmid w$
 12. $v_i \leftarrow (w^{-1} \bmod q^{\alpha_q}) \cdot \frac{e_{i-1}}{q^\beta} \bmod q^{\alpha_q}$
 13. $g_i \leftarrow b_i \quad e_i \leftarrow v_i$
 14. Output v_i
-

Now, [Algorithm 4.4](#) gives rise to a method of computing f_g . This is asserted by the below theorem.

THEOREM 4.4. *There exists a deterministic online algorithm that given a sequence $\alpha_1, \dots, \alpha_\ell \in (\mathbb{Z}/p\mathbb{Z})^*$ computes $f_g(\alpha_1), \dots, f_g(\alpha_\ell)$, where f_g is an epimorphism defined by (3.2) for some, a priori unknown, primitive element g of $\mathbb{Z}/p\mathbb{Z}$ depending on $\alpha_1, \dots, \alpha_\ell$. The algorithm outputs $f_g(\alpha_i)$ for $i = 1, \dots, \ell$ before reading a subsequent argument α_{i+1} . A single value $f_g(\alpha_i)$ can be found in $O(P^+(M) \cdot s^{-1} \text{poly}(\log p))$ time using $O(s \text{poly}(\log p))$ bits of space, where $0 < s \leq \sqrt{P^+(M)}$ is arbitrary.*

PROOF. Assume that **Algorithm 4.4**, having received a sequence $\alpha_1, \dots, \alpha_\ell$, outputs v_1, \dots, v_ℓ . First, we demonstrate that there exists an isomorphism $\psi_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$ satisfying

$$\psi_q(\alpha_i^{(p-1)/q^{\alpha_q}}) = v_i.$$

Clearly, such ψ_q does not necessarily need to be unique.

An immediate observation is that for each b_i computed in Line 4 of **Algorithm 4.4**, it holds that $\text{ord}(b_i) \mid q^{\alpha_q}$, i.e., $b_i \in G_q$. The loop of Lines 3–14 preserves the following invariant: g_i is a generator of $\langle b_0, b_1, \dots, b_i \rangle \subseteq G_q$. Clearly, $\langle g_0 \rangle = \langle b_0 \rangle$. Now, suppose that $\langle g_i \rangle = \langle b_0, b_1, \dots, b_i \rangle$ after the i th iteration. If $\text{ord}(b_{i+1}) \leq \text{ord}(g_i)$ then $b_{i+1} \in \langle g_i \rangle$, and setting $g_{i+1} := g_i$ does not violate the invariant. In the case when $\text{ord}(b_{i+1}) > \text{ord}(g_i)$, we get $g_i \in \langle b_{i+1} \rangle$ so $\langle b_{i+1} \rangle = \langle b_0, b_1, \dots, b_{i+1} \rangle$, and again the invariant is preserved by letting $g_{i+1} := b_{i+1}$.

We claim that **Algorithm 4.4** constructs a sequence of implicit monomorphisms $\mu_i: \langle g_i \rangle \hookrightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$ induced by $\mu_i(g_i) := e_i$ with a property that μ_i restricted to $\langle g_{i-1} \rangle$ is μ_{i-1} . Letting $e_0 := q^{\alpha_q}/\text{ord}(g_0)$ fixes a monomorphism $\mu_0: g_0 \mapsto e_0$ which is extended afterwards, each time the *else* branch of Lines 9–13 is executed by the algorithm and the generator actually changes. We

first note that when $g_{i-1} \in \langle b_i \rangle$ then $q^\beta \mid e_{i-1}$, where $u = q^\beta w$, $q \nmid w$, and $u = \log_{b_i} g_{i-1}$ was computed in Line 10. Indeed,

$$\begin{aligned} q^{\alpha_q - \beta} e_{i-1} &= q^{\alpha_q - \beta} \mu_{i-1}(g_{i-1}) = \mu_{i-1}(g_{i-1}^{q^{\alpha_q - \beta}}) \\ &= \mu_{i-1}(b_i^{wq^{\alpha_q}}) = \mu_{i-1}(1) = 0 \pmod{q^{\alpha_q}}. \end{aligned}$$

As the result, the division in Line 12 can be performed over integers. A simple calculation confirms that restricting μ_i to $\langle g_{i-1} \rangle$ yields μ_{i-1} :

$$\begin{aligned} \mu_i(g_{i-1}) &= \mu_i(b_i^u) = u \cdot e_i \\ &= q^\beta \cdot w \cdot (w^{-1} \pmod{q^{\alpha_q}}) \cdot e_{i-1} / q^\beta \pmod{q^{\alpha_q}} \\ &= e_{i-1} = \mu_{i-1}(g_{i-1}). \end{aligned}$$

We also observe that the mapping $g_i \mapsto e_i$ is order preserving, i.e., $\text{ord}(g_i) = \text{ord}(e_i)$ for each $i = 1, \dots, \ell$. This follows from the fact that $\frac{q^{\alpha_q}}{\text{ord}(b_i)} \parallel e_i$, which can be proven inductively using the formula deriving e_i from e_{i-1} . This property of μ_i implies that μ_i is injective.

Now, if after the ℓ th iteration we end up with $\langle b_\ell \rangle \subsetneq G_q$, then there exists an extension of μ_ℓ to an isomorphism $G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$, which could be computed by repeating Lines 9–13 with additional $b_{\ell+1}$ being a generator of G_q . It is also clear that $\mu_i(b_i) = v_i$ for each i . This proves the existence of ψ_q for every $q^{\alpha_q} \parallel M$. By Lemma 4.1 and Lemma 4.2, one can efficiently obtain $f_g(a_i)$ from $\psi_q(a_i^{(p-1)/q^{\alpha_q}})$ via the CRT.

We note that changing the initial value b_0 , set in Line 2 of Algorithm 4.4, may lead to a different base g of f_g .

For a single q , each b_i and g_i belongs to G_q , and so the cost of order comparison is $O(\text{poly}(\log p))$. Therefore, the most time and space consuming step is determining discrete logarithms in Lines 6 and 10. Employing the PH algorithm together with the BSGS method allows achieving it in $O(qs^{-1} \text{poly}(\log p))$ time and

$O(s \text{ poly}(\log p))$ space, where $0 < s \leq q^{1/2}$ can be chosen in advance. The overall complexity can be bounded above, as stated in the theorem, by the cost of [Algorithm 4.4](#) for $q = P^+(M)$ times some polylogarithmic factor. \square

INDISTINGUISHABILITY

A notable difference between a non-malleable extractor and our algorithm from [Theorem 4.5](#) is that the former is defined in terms of statistical distance between two distributions. Clearly, this interpretation becomes void for distributions induced by a single output of the algorithm, which is supposed to work for multiple arguments. Although our construction does not constitute a replacement for the non-malleable extractor [\(3.4\)](#) in classic scenarios, e.g., the privacy amplification protocol analyzed by [Dodis & Wichs \(2009\)](#), we can still conceive of some analogue of indistinguishability that applies to this algorithm.

COROLLARY 4.5. *Take p , M , k , and $\epsilon = 2Mp^{1/4}2^{-k/2}$ as in [Theorem 3.3](#). Let X and Y be two independent random variables distributed over $\mathbb{Z}/p\mathbb{Z}$ with $H_\infty(X) = k$ and Y being uniform on $\mathbb{Z}/p\mathbb{Z}$. Consider any, possibly computationally unbounded, distinguisher which is allowed to pick a fix-point free function $\mathcal{A}: \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{Z}/p\mathbb{Z}$. Suppose that the following experiment is run:*

(i) *sample $\ell \geq 4$ triples $(x_i, y_i, u_i)_{i=1, \dots, \ell}$, where x_i is chosen from the distribution X , y_i from Y , u_i from $\mathcal{U}_{\mathbb{Z}/M\mathbb{Z}}$, and each element is sampled independently of all other choices,*

(ii) *run the algorithm from [Theorem 4.5](#) on the sequence*

$$x_1 + y_1, x_1 + \mathcal{A}(y_1), \dots, x_\ell + y_\ell, x_\ell + \mathcal{A}(y_\ell)$$

of length 2ℓ obtaining

$$v_1, v'_1, \dots, v_\ell, v'_\ell$$

as the result,

(iii) the distinguisher is given ℓ challenges of the form: (v_i, v'_i, y_i) and (u_i, v'_i, y_i) simultaneously.

Then, for $i = 4, \dots, \ell$ the probability (taken over random coin tosses of the distinguisher and all random choices made during the experiment when sampling $x_i, y_i,$ and u_i) that the distinguisher tells apart the i th pair (v_i, v'_i, y_i) and (u_i, v'_i, y_i) does not exceed $1/2 + \epsilon + 6 \cdot 2^{-i}$.

PROOF. We relate these chances to the ones of any distinguisher for the extractor of [Theorem 3.3](#), where a correct guess is made with probability at most $1/2 + \epsilon$, uniformly for each pair. Ideally, we would like to view each realization in our experiment as an independent challenge. A possible dependence of algorithm's output on previous inputs (due to the relationship between g_i and b_j for $j = 1, \dots, i$ in [Algorithm 4.4](#)) is thus an issue. That said, from the moment when for each $q \mid M$ the corresponding b_i generates the whole group G_q , only the first branch in Lines 6-8 of If-Else statement in [Algorithm 4.4](#) gets executed. For a fixed generator of G_q there is a fixed isomorphism ψ_q these lines implement verbatim. Put differently, if $\{(x_j + y_j)^{(p-1)/M} \mid 1 \leq j < i\}$ is a generating set of G then there is no hidden dependence between subsequent challenges starting from the i th one. The only things that the distinguisher can learn from the initial $i - 1$ challenges are: the isomorphism ψ induced by ψ_q with a set of primitive elements g satisfying $g^{(p-1)/M} = \psi^{-1}(1)$ (cf. [Lemma 4.1](#)), the distributions X and Y . All these parameters are "public" and do not grant the distinguisher any advantage, which follows from [Theorem 3.3](#).

Now, we note that since each y_j for $j = 1, \dots, i - 1$ is sampled uniformly and independently, the values $z_j = (x_j + y_j)^{(p-1)/M}$ are equidistributed over G . We argue that there may be only a short sequence of initial distinguisher's guesses which are correct solely because $\langle z_1, \dots, z_{i-1} \rangle \neq G$. In fact, a general result by [Pomerance \(2002\)](#) shows that the expected number of random samples to generate any finite cyclic group is less than three. Reducing his argument to our particular case we let $P_j = \prod_{q \mid M} (1 - q^{-(j+1)})$ to

be the probability that $j + 1$ random elements of G jointly generate the whole group. The probability of an event that the distinguisher tells apart the i th pair conditioned on the event that $\langle z_1, \dots, z_{i-1} \rangle = G$ is simply $\leq 1/2 + \epsilon$. It remains to estimate $1 - P_{i-2}$. Bounding P_j above by Euler product for the Riemann ζ function we get that $P_j \geq \zeta(j+1)^{-1}$ for $j \geq 1$. Since $\zeta(j) \leq 1 + 2^{-j} + \int_2^\infty t^{-j} dt \leq 1 + 3 \cdot 2^{-j}$ for $j \geq 2$ we have $1 - P_{i-2} \leq 1 - 1/(1 + 3 \cdot 2^{-i+1}) \leq 6 \cdot 2^{-i}$. \square

We also notice that the additional $6 \cdot 2^{-i}$ term in adversary's advantage actually arises only in steps where the currently known subgroup $\langle g_i \rangle$, considered in [Theorem 4.5](#), grows after adding a new input. In the other case the probability of making a correct guess is at most $1/2 + \epsilon$ as guaranteed by [Theorem 3.3](#). As the final remark, we note that $P_j \geq \prod_{q \leq M} (1 - q^{-(j+1)})$, where the product is taken over all primes $q \leq M$. Using a generalization of Mertens' theorem for such a product, one can bound the additional error term slightly better. Namely, we can hope for $1/\log M$ factor that further reduces this term yet this is only a minor improvement.

NON-MALLEABLE EXTRACTOR WITHOUT THE ERH

In this chapter, we provide an alternative construction of a non-malleable extractor. As announced in [Chapter 3](#), our idea is to replace discrete logarithm appearing in the Chor-Goldreich extractor with exponentiation. This way we circumvent the problem of finding a primitive element g of $\mathbb{Z}/p\mathbb{Z}$ and remove a possible dependence on the ERH. We still need a prime p and $M \mid p - 1$, although in this chapter it is not required to assume that M is smooth. First, we describe our extractor, which is effective provided that both parameters are given, and then we reassess possibilities of generating suitable numbers p and M .

LEMMA 5.1. *Let p , M , k , and ϵ be constrained in the same way as in [Theorem 3.3](#). Then, the function $\text{Ext}_G: \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \rightarrow G$ given by*

$$\text{Ext}_G(x, y) := (x + y)^{(p-1)/M}$$

is an efficiently computable (k, ϵ) -non-malleable extractor.

PROOF. Set g to be any primitive element of $\mathbb{Z}/p\mathbb{Z}$. Then,

$$\langle g^{(p-1)/M} \rangle = G .$$

Fix an isomorphism $\psi: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ induced by $\psi(g^{(p-1)/M}) = 1$. By [Lemma 4.1](#) for all $x, y \in \mathbb{Z}/p\mathbb{Z}$ it holds that

$$\text{Ext}(x, y) = \psi(\text{Ext}_G(x, y)) ,$$

where Ext is the non-malleable extractor from [Theorem 3.3](#) based on f_g . That is, the relation [\(3.1\)](#) holds for Ext and any independent variables X with $H_\infty(X) \geq k$ and Y – uniformly random on $\mathbb{Z}/p\mathbb{Z}$.

Since ψ is bijective then by rearranging terms in the formula for statistical distance between

$$(\text{Ext}(X, Y), \text{Ext}(X, A(Y)), Y) \quad \text{and} \quad (U_{\mathbb{Z}/M\mathbb{Z}}, \text{Ext}(X, A(Y)), Y)$$

we get that the distance is equal to the one between

$$(\psi^{-1} \circ \text{Ext}(X, Y), \psi^{-1} \circ \text{Ext}(X, A(Y)), Y) \quad \text{and} \\ (U_G, \psi^{-1} \circ \text{Ext}(X, A(Y)), Y) .$$

Therefore, Ext_G is a non-malleable extractor with the same error ϵ as Ext and can be computed in $O(\text{poly}(\log p))$ time. \square

In order to set up the extractor, i.e., to fix M and p , defined in the above lemma, we can follow one of the approaches outlined in [Chapter 3](#). For instance, under [Conjecture 3.5](#) this can be done deterministically for a wide range of M . Unconditionally, if we keep to [Chor & Goldreich](#) then [Theorem 3.8](#) leads to an expected polynomial time algorithm and only superpolynomial ϵ^{-1} . An exponential bound is achievable via [Theorem 3.6](#) for extractors with short output, and presumably for all plausible M assuming the ERH. We show how to make such a transition to large M without relying on any conjectures. This stems directly from the fact that the error term in [\(3.7\)](#) can be reliably controlled even for large (exceeding the polylogarithmic bound) d with only few exceptional moduli d . Of several results of this type, among them the Bombieri-Vinogradov (BV) theorem, we apply the one from the influential paper by [Alford et al. \(1994\)](#), which is best tailored to our particular application.

THEOREM 5.2 ([Alford et al. 1994](#)). *For every $\kappa > 0$ and $0 < \lambda < \frac{5}{12}$ there exist constants $\gamma > 0$, x_0 , and D , all depending only on κ and λ , such that for every $x \geq x_0$ there is a set $\mathcal{D}(x)$ of at most D integers with the following property: for every y and each $1 \leq d \leq \min(x^\lambda, y/x^{1-\lambda})$ coprime to a and not divisible by any element of $\mathcal{D}(x)$ it holds that*

$$\left| \sum_{p \leq y}^* \log p - \frac{y}{\varphi(d)} \right| \leq \kappa \frac{y}{\varphi(d)},$$

where \sum^* runs over all primes $p \equiv a \pmod{d}$ and a is fixed. Moreover, all elements of $\mathcal{D}(x)$ exceed $\log x$, and all, but at most one, exceed x^γ .

As an immediate consequence of [Theorem 5.2](#) we obtain the below corollary.

COROLLARY 5.3. *For all sufficiently large z, z' , and all $l > 0$ satisfying $(z+l)^4 < \frac{1}{2}z'$ there exist at least $\frac{1}{3}z'/(\varphi(d) \log z')$ primes in the interval $(\frac{1}{2}z', z']$ that belong to the arithmetic progression $1 \pmod{d}$ for every modulus d from $(z, z+l)$ with at most $O(l/\log(z+l))$ exceptional d .*

PROOF. Fix $\kappa := \frac{1}{9}$, $\lambda := \frac{1}{4}$, and apply [Theorem 5.2](#) with $x := (z+l)^4$, $y := z'$, provided that z is sufficiently large. Then the interval $(z, z+l)$ is contained in the range of valid d the theorem is applicable to. Ignoring possible overlaps, there are at most $D \cdot (\lfloor l/\log x \rfloor + 1) = O(l/\log(z+l))$ integers in $(z, z+l)$ divisible by some element from the set $\mathcal{D}(x)$. For every other d in this interval we have that

$$\sum_{p \leq z'}^* \log p \geq (1 - \kappa)z'/\varphi(d).$$

Applying [Theorem 5.2](#) the second time with y replaced with $y := \frac{1}{2}z'$ yields, for the same modulus d , an upper bound of the form

$$\sum_{p \leq \frac{1}{2}z'}^* \log p \leq \frac{1}{2}(1 + \kappa)z'/\varphi(d).$$

Combining these two estimates together and bounding $\log p$ trivially by $\log z'$ we get the claimed result. \square

Now, suppose that we are to construct a (k, ϵ) -non-malleable extractor being given a min-entropy rate $1/2 + \delta \leq 1$, an error ϵ , and a number of bits m the extractor should output. These parameters can be realized if, for some m -bit modulus M , we choose $p \equiv 1 \pmod{M}$ to be an n -bit prime, where

$$n := \lceil 2\delta^{-1}(\log_2 \epsilon^{-1} + m + 2) \rceil.$$

The last value follows from the error bound $\epsilon = 2Mp^{1/4}2^{-k/2}$ of [Theorem 3.3](#) with $k = (1/2 + \delta) \log_2 p$. Under these constraints we pick pairs (M, p) in a nondeterministic way, sampling M first and then testing whether a random element p in the arithmetic progression mod M is prime. By [Corollary 5.3](#) the expected number of Bernoulli trials until the first success does not exceed

$$(1 - O(1/m))^{-1} \cdot O(\log 2^{n+1}) = O(n) .$$

Alternatively, we can derive a similar corollary from the better known BV theorem, which provides even more relaxed condition on d , yet at the expense of lengthening intervals where d or p should be searched in, and, more importantly, of having ineffective constants (due to appeal to the Siegel-Walfisz theorem). Since the formula for ϵ implies that we are only interested in the case where $p > M^4$, the BV theorem offers no advantage. On the other hand, the proof method used by [Alford *et al.*](#) does allow for computing all the constants referenced in [Theorem 5.2](#), and thus the estimate of running time of our algorithm is effective.

EFFECTIVE BIJECTION

A drawback of the solution from [Chapter 5](#) is that Ext_G produces values belonging to the group G . It may be regarded impractical as G , viewed as a set of bit-strings, has an “irregular” structure. Thus, one may favor an extractor outputting values from $\mathbb{Z}/M\mathbb{Z}$, which are easily transformable to nearly random bits, over Ext_G . Below, we work out an explicit bijective mapping $\sigma: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ so that $\sigma \circ \text{Ext}_G$ is such a preferred extractor. The construction implies that σ is efficiently computable when M is a smooth number, which essentially means that calculating discrete logarithms in G is feasible. We, however, do not assume that a generator of G is known a priori nor we attempt to find one nondeterministically. Our result can be easily generalized to any cyclic group of smooth order, not necessarily a multiplicative subgroup of a prime field, as long as its elements possess a natural representation as bit-strings.

We note that the problem of finding efficiently computable σ for general groups of any order may be of independent interest. It could be of practical value if achieved for cryptographically significant groups, where the DLP is actually hard. For instance, when used as a key derivation primitive after Diffie-Hellman key exchange phase it would allow extracting a random string from a random group element. [Chevassut *et al.* \(2006\)](#) give a simple extractor for that when $p = 2q + 1$ and q is an odd Sophie Germain prime. Then, taking G to be the subgroup of quadratic residues mod p , letting $M = q$, and interpreting elements of G as integers, we can define the bijection by:

$$\sigma(a) = \begin{cases} a & \text{if } a \leq q \\ p - a & \text{otherwise.} \end{cases}$$

Sophie Germain primes often appear in handbook cryptographic protocols yet their density remains conjectured, and it is not even known whether there exist infinitely many such numbers. Recently, Joachim von zur Gathen has pointed out to us that safe primes p of slightly more general form, namely such that $(p - 1)/2$ possesses at most two prime factors and each factor is large, are provably sufficiently dense and can be reliably generated (von zur Gathen & Shparlinski 2013). As every number outputted by their algorithm is a Blum prime, i.e., $p \equiv 3 \pmod{4}$, and thus -1 is a quadratic non-residue modulo p , the above extractor by Chevassut *et al.* also applies to such primes. However, the smoothness condition we assume makes constructing σ possible for much wider family of groups. To the best of our knowledge, the case where no generator of G is given has not been considered before.

In the first step, we describe a method for calculating a bijection $\sigma_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$ for each $q^{\alpha_q} \parallel M$. The big picture of how the construction of σ_q looks like is quite simple: we organize members of G_q in a certain sequence and let σ_q correspond to a position in that sequence. The elements are arranged with respect to their multiplicative orders in $(\mathbb{Z}/p\mathbb{Z})^*$. If these happen to be the same for some two elements then we recursively check positions of q th powers of these elements. We formalize this approach below.

Fix a prime divisor q of M . For any $a \in G_q$ we define the *radical* $R(a)$ of a to be $R(a) := \{b \in G_q \mid b^q = a^q\}$. Suppose that we are given an auxiliary function $\theta_q: G_q \rightarrow \{0, 1, \dots, q - 1\}$ with $\theta_q(1) = 0$ and satisfying the following property:

for each $a \in G_q$ the function θ_q restricted to the radical $R(a)$ is a bijective mapping $R(a) \rightarrow \{0, 1, \dots, q - 1\}$.

Below, we give two examples of how θ_q can be specified in a way enabling its efficient evaluations provided M is smooth.

Based on θ_q we define a strict total order \prec on G_q recursively so that $b \prec a$ iff $b \neq a$ and one of the below conditions holds:


$$(6.1) \quad \text{ord}(b) < \text{ord}(a)$$

$$(6.2) \quad \text{ord}(b) = \text{ord}(a) \text{ and } b^q \prec a^q$$

$$(6.3) \quad b^q = a^q \text{ and } \theta_q(b) < \theta_q(a).$$

Finally, we let $\sigma_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$ be a bijective mapping induced by this ordering:

$$(6.4) \quad \sigma_q(a) := |\{b \in G_q \mid b \prec a\}|.$$

Figure 6.5 depicts how this construction works. Each cell represents a single element of G_q . All q elements of every radical are grouped as  except for the neutral element of G_q , which is drawn separately.

Our intermediate goal is to prove that if θ_q is efficiently computable then so is σ_q .

LEMMA 6.6. *The function σ_q satisfies the following recursive relation*

$$(6.7) \quad \sigma_q(a) = \begin{cases} \theta_q(a) & \text{if } \text{ord}(a) \leq q \\ q \cdot \sigma_q(a^q) + \theta_q(a) & \text{if } \text{ord}(a) \geq q^2. \end{cases}$$

PROOF. By the definition of \prec , the claim is clear for $a \in G_q$ with $\text{ord}(a) \leq q$. We verify the second property of (6.7). Let $\text{ord}(a) = q^\alpha$ for $\alpha \geq 2$. Consider a partition of $\{b \in G_q \mid b \prec a\}$ into sets \mathcal{S}_1 , \mathcal{S}_2 , and \mathcal{S}_3 implied by (6.1)-(6.3):

$$\mathcal{S}_1 := \{b \in G_q \mid \text{ord}(b) < q^\alpha\}$$

$$\mathcal{S}_2 := \{b \in G_q \mid \text{ord}(b) = q^\alpha \text{ and } b^q \prec a^q\}$$

$$\mathcal{S}_3 := \{b \in R(a) \mid \theta_q(b) < \theta_q(a)\}.$$

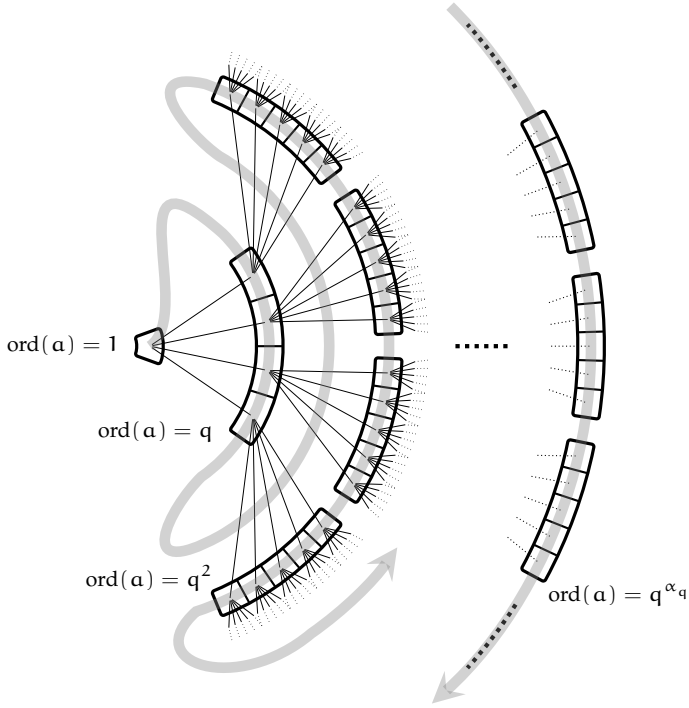


Figure 6.5: An ordering of G_q for $q = 5$ inducing a bijection $\sigma_q: G_q \rightarrow \mathbb{Z}/q^{\alpha_q}\mathbb{Z}$.

It is straightforward that $|\mathcal{S}_1| = q^{\alpha-1}$, $|\mathcal{S}_3| = \theta_q(a)$, and

$$\begin{aligned} |\mathcal{S}_2| &= q \cdot |\{c \in G_q \mid \text{ord}(c) = q^{\alpha-1} \text{ and } c \prec a^q\}| \\ &= q \cdot (|\{c \in G_q \mid c \prec a^q\}| - |\{c \in G_q \mid \text{ord}(c) < q^{\alpha-1}\}|) \\ &= q \cdot (\sigma_q(a^q) - q^{\alpha-2}). \end{aligned}$$

Hence, $\sigma_q(a) = |\mathcal{S}_1| + |\mathcal{S}_2| + |\mathcal{S}_3| = q \cdot \sigma_q(a^q) + \theta_q(a)$. □

We are now ready to prove the principal result of this section.

THEOREM 6.8. *There exists a bijective mapping $\sigma: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ such that for a given $h \in G$ the corresponding value $\sigma(h)$ can be computed by a deterministic algorithm in $O(P^+(M) \text{poly}(\log p))$ time.*

PROOF. For the sake of this proof we instantiate θ_q with a bijection induced by lexicographical order on each radical. That is:

$$\theta_q(a) := |\{b \in R(a) \mid b < a\}|,$$

where the comparison $b < a$ is over integers. We argue that the bijection σ_q given by (6.4) for every $q \mid M$ can be computed efficiently, i.e., in $O(q \text{poly}(\log p))$ steps, using the dependency of Lemma 6.6. Take any $a \in G_q$. We can assume that $a \neq 1$ and let $\text{ord}(a) = q^\alpha$ for some $\alpha \geq 1$. To find $\sigma_q(a)$, by (6.7), we need to compute the successive values $\sigma_q(a^{q^{\alpha-1}}), \sigma_q(a^{q^{\alpha-2}}), \dots, \sigma_q(a)$, and the only difficulty lies in determining $\theta_q(a^{q^i})$ for each $i = 0, \dots, \alpha - 1$. Note that for any $c \in G_q \setminus \{1\}$ the radical $R(c)$ is composed of precisely these $b_j \in G_q$ which can be expressed as $b_j = \varepsilon^j \cdot c$, where ε is some q th root of unity in G_q , and $j = 0, \dots, q - 1$. For instance, we can take $\varepsilon = a^{q^{\alpha-1}}$. All b_j are pairwise distinct. Therefore

$$R(a^{q^i}) = \{(a^{q^{\alpha-1}})^j \cdot a^{q^i} \mid 0 \leq j \leq q - 1\}.$$

Calculating $\theta_q(a^{q^i})$ is thus a straightforward task – it suffices to generate all members of $R(a^{q^i})$ on the fly and count only these elements that precede a^{q^i} lexicographically. Here, we observe that the whole process can be achieved within $O(\text{poly}(\log p))$ space.

Applying Lemma 4.2 we obtain a bijection σ such that a single value of σ is computable via the CRT in $O(P^+(M) \text{poly}(\log p))$ deterministic time. \square

We are also able to devise an algorithm that calculates a bijection for multiple arguments reducing the asymptotic cost for a single computation.

ALGORITHM 2. Online computing of θ_q .

Input: A sequence $a_1, \dots, a_\ell \in G_q \setminus \{1\}$

Output: A sequence $\theta_q(a_1), \dots, \theta_q(a_\ell)$

1. $\alpha \leftarrow 0 \quad \varepsilon_\alpha \leftarrow 1$
 2. For $i \leftarrow 1, \dots, \ell$ do 3–10
 3. find β such that $\text{ord}(a_i) = q^\beta$
 4. While $\beta > \alpha$ do 5–6
 5. compute $0 < u \leq q^\beta$ such that $a_i^u = \varepsilon_\alpha$
 6. $\alpha \leftarrow \alpha + 1 \quad \varepsilon_\alpha \leftarrow \max_{\leq} R(a_i^{u/q})$
 7. $\varepsilon_\beta \leftarrow \varepsilon_\alpha^{q^{\alpha-\beta}}$
 8. compute $0 \leq t < q^{\beta-1}$ such that $(\varepsilon_\beta^q)^t = a_i^q$
 9. compute $0 \leq v < q$ such that $(\varepsilon_\beta^{q^{\beta-1}})^v = a_i \varepsilon_\beta^{-t}$
 10. Output v
-

With [Algorithm 6.9](#) we can prove the following result.

THEOREM 6.9. *There exists a deterministic online algorithm that computes some fixed bijective mapping $G \rightarrow \mathbb{Z}/M\mathbb{Z}$. The algorithm uses $O(s \text{ poly}(\log p))$ bits of memory, and its total running time for ℓ inputs is $O(P^+(M)(\ell s^{-1} + 1) \text{ poly}(\log p))$, where $0 < s \leq \sqrt{P^+(M)}$ is arbitrary.*

PROOF. By [Lemma 4.2](#) and [Lemma 6.6](#) it suffices to show that, for each $q^{\alpha_q} \parallel M$, [Algorithm 6.9](#), given $a_1, \dots, a_\ell \in G_q \setminus \{1\}$, outputs $\theta_q(a_1), \dots, \theta_q(a_\ell)$ for certain θ_q , and that it does so within $O(q(\ell s^{-1} + 1) \text{ poly}(\log p))$ time using $O(s \text{ poly}(\log p))$ space. First, we provide an explicit formula for θ_q . With each order q^α for $\alpha = 0, \dots, \alpha_q$ we associate an element $\varepsilon_\alpha \in G_q$ such that $\text{ord}(\varepsilon_\alpha) = q^\alpha$, and ε_α is defined inductively by: $\varepsilon_0 := 1$ and

$$(6.10) \quad \varepsilon_{\alpha+1} := \max_{\leq} \{b \in G_q \setminus \{1\} \mid b^q = \varepsilon_\alpha\},$$

where, again, \leq is the lexicographical order. Also, we pair every $a \in G_q \setminus \{1\}$ with $\xi = \xi(a) \in R(a)$ in the following way:

let $\text{ord}(a) = q^\beta$ and t be the smallest positive exponent satisfying $\varepsilon_{\beta-1}^t = a^q$. Then $\xi(a) := \varepsilon_\beta^t$. One can easily verify that $\xi^q = \varepsilon_\beta^{tq} = \varepsilon_{\beta-1}^t = a^q$ so, by the definition of radical, $\xi \in R(a)$. Finally, we set

$$(6.11) \quad \theta_q(a) := \log_\varepsilon(a\xi^{-1}) \quad \text{for } a \neq 1,$$

where $\varepsilon := \varepsilon_1$ and $\xi := \xi(a)$. This is a well-defined function admitting values from $\{0, 1, \dots, q-1\}$, which comes from the fact that a and ξ belong to the same radical, $a\xi^{-1}$ and ε are q th roots of unity in G_q . Further, observe that all elements of $R(a)$ share the same t , and thus $\xi(a) = \xi(a')$ for every $a' \in R(a)$, $a' \neq 1$. Therefore θ_q restricted to each $R(a)$ is injective, so indeed it satisfies the required property.

Now, for each $a = a_i$ of order q^β , provided that ε_β is calculated correctly in Line 7 of Algorithm 6.9, Lines 8 and 9 literally implement (6.12) because $\varepsilon_\beta^q = \varepsilon_{\beta-1}$ and $\varepsilon_\beta^{q^{\beta-1}} = \varepsilon$. If, at the beginning of the i th iteration of Lines 2–10, we happen to have $\beta \leq \alpha$ then, clearly, ε_β is the $q^{\alpha-\beta}$ th power of current ε_α . In the other case, a_i is of larger order than ε_α , so the exponent u calculated in Line 5 is divisible by q . It is thus possible to update ε_α according to (6.11) and advance α up to β .

The use of the PH algorithm with the BSGS method to find u , t , and v contributes $O(q/s \text{ poly}(\log p))$ to time complexity in each iteration of the main loop, where $s \leq \sqrt{P^+(M)}$ can be chosen arbitrarily. Each update of ε_α in Line 6 requires enumerating all elements of the radical $R(a_i^{u/q})$, which can be achieved, as we have already mentioned in the proof of Theorem 6.8, in $O(q \text{ poly}(\log p))$ steps using some q th root of unity, e.g., $a_i^{q^{\beta-1}}$. The overall number of times the inner loop can be executed is at most $\alpha_q = O(\log p)$. Hence the total cost of the algorithm. \square

We note that in the above proof the functions θ_q and, consequently, the induced bijection are all fixed, i.e., they do not depend on inputs supplied to the algorithm, as opposed to the construc-

tion from [Theorem 4.5](#). Also, for $\ell = \Omega(s)$ the time complexity becomes $O(\ell \cdot P^+(M) \cdot s^{-1} \text{poly}(\log p))$ as we can disregard the term that emerges from finding members of each radical $R(a_i^{u/q})$. Thus, the amortized cost for a single argument keeps up with the standard “ $\frac{d}{s}$ -time vs. s -memory” correspondence of the BSGS method for groups of order q .

NON-MALLEABLE EXTRACTOR WITH LONG OUTPUT

Below, we complete the development of the non-malleable extractor initiated in [Chapter 5](#). The bijection $\sigma: G \rightarrow \mathbb{Z}/M\mathbb{Z}$ constructed in [Chapter 6](#) provides a lossless conversion of our extractor's output to $\mathbb{Z}/M\mathbb{Z}$. That is, $\sigma \circ \text{Ext}_G$ achieves the same error ϵ as Ext_G has. The computational limitation, however, is that σ remains effective only when M is smooth. It essentially means that we end up with the same problem concerning M and p as in the extractor by [Dodis *et al.*](#) discussed in [Chapter 3](#). The procedure we have for generating these parameters, but with an additional smoothness requirement on M , is conditional unless M is small, polynomial in $\log p$. Now, we focus on the more challenging case of a non-malleable extractor with long output, i.e., having a range of cardinality $p^{\Omega(1)}$.

An elementary observation towards our final construction is that we can, in fact, allow some entropy loss when transforming elements of G to strings of bits. Put differently, we do not necessarily need σ in the composition $\sigma \circ \text{Ext}_G$ to be injective. The solution we propose is to take some fixed fraction of bits appearing in bit representation of elements of G . This is a natural and quite simple approach, yet a strict estimate of error bound for such an extractor is not that straightforward and requires more involved arguments.

The idea of restricting representations to least significant bits is not new in the theory of randomness extractors. For instance, one of the classical examples in this field (cf. [Holenstein \(2006\)](#)) uses trimming elements of binary Galois fields as its ingredient. [Fouque *et al.* \(2006\)](#) and [Chevalier *et al.* \(2009\)](#) study suitability of such a method for extracting random bits from Diffie-Hellman elements of multiplicative groups of prime fields and of groups on

elliptic curves over prime fields. Finally, [Dodis *et al.*](#) also suggest a similar way to circumvent the inconvenient divisibility condition: $M \mid p - 1$. We briefly restate their result to compare the parameters they attain with ours.

Let p and $N < p$ be given numbers, and g be a primitive element of $\mathbb{Z}/p\mathbb{Z}$. Define a function $\sigma: \mathbb{Z}/(p-1)\mathbb{Z} \rightarrow \mathbb{Z}/N\mathbb{Z}$ by $\sigma(a) := a \bmod N$. [Dodis *et al.*](#) consider an extractor

$$(7.1) \quad \text{Ext}'(x, y) := \sigma(\log_g(x + y)) = \log_g(x + y) \bmod N,$$

which is given by virtually the same formula as Ext before, yet this time we allow that $N \nmid p - 1$. Clearly, this construction is purely theoretical because evaluating Ext' gets even more intricate. Namely, no more efficient method to do this is known other than finding $\log_g(x + y)$ first and then reducing the result $\bmod N$. Using the PH algorithm for this purpose yields running time roughly proportional to $P^+(p - 1)$ which is not practical.

Now, for a character ϕ of $\mathbb{Z}/N\mathbb{Z}$ the composition $\phi \circ \sigma \circ \log_g$ may not be a Dirichlet character of $(\mathbb{Z}/p\mathbb{Z})^*$ if $N \nmid p - 1$. This is a problem which impedes proving that for $N \approx M$ the above extractor Ext' enjoys approximately the same error bound as Ext does. The expectation (2.2) cannot be estimated with Weil's bound, and thus the hypothesis of [Lemma 2.1](#) is not satisfied. Instead, [Dodis *et al.*](#) establish the following extension of the non-uniform XOR lemma.

LEMMA 7.2 ([Dodis *et al.*](#), Lemma A.3). *Let H_1 and H_2 be finite abelian groups. Suppose that for two random variables Z and Z' , distributed over H_1 , it holds that $|\mathbb{E}_{(Z, Z')}[\phi(Z)\phi'(Z')]| \leq \alpha$ for every pair ϕ, ϕ' of characters over H_1 with ϕ non-trivial. Let $\sigma: H_1 \rightarrow H_2$ be any function satisfying*

$$(7.3) \quad \|\widehat{\psi \circ \sigma}\|_{\ell_1} \leq \beta \cdot |H_1|$$

for every character ψ of H_2 . Then,

$$\Delta\left((\sigma(Z), \sigma(Z')), (\sigma(U_{H_1}), \sigma(Z'))\right) \leq \frac{1}{2} \alpha \beta \cdot |H_1|,$$

where U_{H_1} is uniform and independent of Z' .

In a sense, the condition (7.3) captures the distance between σ and a homomorphism. We note that if σ is an epimorphism then, for a non-trivial ψ , the composition $\psi \circ \sigma$ is also a non-trivial character. In this scenario the left-hand side of (7.3) is equal to $|H_1|$ and Lemma 7.2 becomes the same as Lemma 2.1.

To bound the ℓ_1 -norm of $\widehat{\psi \circ \sigma}$ in the particular case where σ is a reduction modulo an integer, we can use the following lemma attributed to Rao (2007).

LEMMA 7.4 (Dodis *et al.*, Lemma A.4). *Let L and $N < L$ be integers. For $H_1 = \mathbb{Z}/L\mathbb{Z}$, $H_2 = \mathbb{Z}/N\mathbb{Z}$, a function $\sigma: H_1 \rightarrow H_2$ given by*

$$\sigma(a) := a \bmod N,$$

and any character ψ of H_2 , it holds that $\|\widehat{\psi \circ \sigma}\|_{\ell_1} = O(L \log L)$.

With H_1 , H_2 , and σ as in Lemma 7.4 it is not hard to see that

$$(7.5) \quad \Delta(\sigma(U_{H_1}), U_{H_2}) \leq 2N/L.$$

Now, let X and Y be any independent random variables over $\mathbb{Z}/p\mathbb{Z}$ with $H_\infty(X) \geq k$ and Y uniform on $\mathbb{Z}/p\mathbb{Z}$. Combining Lemma 7.2, Lemma 7.4, Weil's bound for character sums, and (7.5) for $Z = \log_g(X + Y)$, $Z' = \log_g(X + A(Y))$, $L = p - 1$, $H_1 = \mathbb{Z}/L\mathbb{Z}$, and $H_2 = \mathbb{Z}/N\mathbb{Z}$, we get the below result.

THEOREM 7.6 (Dodis *et al.*, Theorem A.1 and Lemma A.2). *For any prime p , a primitive element g of $\mathbb{Z}/p\mathbb{Z}$, and positive integers k and $N < p$, the function Ext' given by (7.1) is a (k, ϵ') -non-malleable extractor with*

$$(7.7) \quad \epsilon' = O(Np^{1/4}2^{-k/2} \log p + N/p).$$

Again, (7.7) is meaningful only for sufficiently large k and moderately large N . Namely, ϵ' is non-trivial if for some constant $C > 0$ we have $k > \frac{1}{2} \log_2 p + \log_2 \log p + C$ and $N \leq 2^{k/2-C} p^{-1/4}$.

Our goal in this chapter is to demonstrate a similar bound but for $\log_g(x + y)$ in (7.1) replaced with $\text{Ext}_G(x, y)$. There are, however, two difficulties on this route. First, Rao's proof of Lemma 7.4 breaks if repeated for $H_1 = G$. This is because his reasoning relies on the fact that elements of H_1 are consecutive integers, which is not the case for G . So instead of using the lemma and Fourier analysis, we bound statistical distance directly at the expense of a slightly worse error estimate. The second problem we encounter pertains to bounding above $\Delta(\sigma(U_G), U_{\mathbb{Z}/N\mathbb{Z}})$ like in (7.5). Obtaining a suitable inequality is possible yet it involves employing more sophisticated tools. We elaborate on this below.

For simplicity, we consider the case where N is a power of two, say $N = 2^n$. Then, the function $\sigma(a) = a \bmod N = \text{lsb}_n(a)$ trims an element $a \in G$ to n least significant bits. Chevalier *et al.* (2009) prove, using exponential sum techniques, that such a truncation of a random element of a subgroup H of $(\mathbb{Z}/p\mathbb{Z})^*$ yields a nearly random string of bits. The authors focus only on rather large groups, i.e., $|H| = \Omega(p^{1/3})$, which have greater cryptographic significance and, more importantly, allow using much sharper estimates for exponential sums over H . We, however, are interested in smaller groups because Ext_G , by Lemma 5.1 and Theorem 3.3, is non-trivial for $M = |G| < p^{1/4}$. Fortunately, we can also cover such groups thanks to the sum-product theorems by Bourgain *et al.*

Write $e_p(x) := e^{\frac{2\pi i}{p}x}$ and let

$$S(H) := \max_{\xi \neq 0} \left| \sum_{h \in H} e_p(h\xi) \right|$$

bound the exponential sum over H . By looking at the proof of Chevalier *et al.* we can infer the following inequality.

LEMMA 7.8 (Chevalier *et al.*, see the proof of Theorem 8). *For any subgroup H of $(\mathbb{Z}/p\mathbb{Z})^*$ it holds that*

$$\Delta(\text{lsb}_n(U_H), U_{\mathbb{Z}/N\mathbb{Z}}) \leq \frac{1}{2} N^{1/2} \cdot (p^{-1/2} + S(H) \cdot |H|^{-1} \log_2^{1/2} p) .$$

It remains to estimate $S(H)$. The well-known Polya-Vinogradov inequality allows taking $S(H) \leq p^{1/2}$ in the above lemma but it is only relevant for $|H| > p^{1/2}$. We apply the below estimate sketched by [Bourgain & Konyagin \(2003\)](#) with a complete proof given by [Bourgain *et al.* \(2006\)](#). Both are based on the powerful result by [Bourgain *et al.* \(2004\)](#).

THEOREM 7.9 ([Bourgain & Konyagin, Theorem 2.1](#)). *There exist positive constants C_1 and C_2 such that for $\beta > 0$ and every subgroup H of order $|H| \geq p^\beta$ we have*

$$S(H) \leq p^{-\gamma}|H|,$$

where $\gamma = 2^{-C_1/\beta^{C_2}}$.

We can now show the following theorem on our extractor.

THEOREM 7.10. *For every constant $\beta > 0$ there exists $\gamma = \gamma(\beta) > 0$ such that for any prime p , an integer $M \mid p-1$ satisfying $M \geq p^\beta$, a min-entropy k , and $n > 0$ the function $\text{Ext}'' : \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{Z}/N\mathbb{Z}$ given by*

$$\text{Ext}''(x, y) := \text{lsb}_n(\text{Ext}_G(x, y)) = \text{lsb}_n((x + y)^{(p-1)/M})$$

is an efficiently computable (k, ϵ'') -non-malleable extractor with

$$\epsilon'' = 2Mp^{1/4}2^{-k/2} + \frac{1}{2}N^{1/2}p^{-\gamma} \log_2^{1/2} p,$$

where $N = 2^n$.

PROOF. Consider any pair X and Y of independent random variables on $\mathbb{Z}/p\mathbb{Z}$, where $H_\infty(X) \geq k$ and Y is uniform. Let Z' and Z'' be shorthands for $\text{Ext}_G(X, Y)$ and $\text{Ext}_G(X, A(Y))$, respectively. Write $Z'_1 = \text{lsb}_n(Z_1)$ and $Z'_2 = \text{lsb}_n(Z_2)$. Let U_G be a uniformly

distributed variable independent of (X, Y) . Then, by the triangle inequality for Δ :

$$(7.11) \quad \Delta((Z'_1, Z'_2, Y), (U_{\mathbb{Z}/N\mathbb{Z}}, Z'_2, Y)) \leq \\ \Delta((Z'_1, Z'_2, Y), (\text{lsb}_n(U_G), Z'_2, Y)) + \\ \Delta((\text{lsb}_n(U_G), Z'_2, Y), (U_{\mathbb{Z}/N\mathbb{Z}}, Z'_2, Y)).$$

Since $\text{lsb}_n: G \rightarrow \mathbb{Z}/N\mathbb{Z}$ is a deterministic function, the first term on the right-hand side of (7.11) can be bounded above by

$$\Delta((Z_1, Z_2, Y), (U_G, Z_2, Y)).$$

By Lemma 5.1 it follows that $(Z_1, Z_2, Y) \approx_\epsilon (U_G, Z_2, Y)$ and so $(Z'_1, Z'_2, Y) \approx_\epsilon (\text{lsb}_n(U_G), Z'_2, Y)$, where $\epsilon = 2Mp^{1/4}2^{-k/2}$ is the error bound from Theorem 3.3. As for the second term of (7.11), it is just $\Delta(\text{lsb}_n(U_G), U_{\mathbb{Z}/N\mathbb{Z}})$ because U_G and $U_{\mathbb{Z}/N\mathbb{Z}}$ are both independent of (Z'_2, Y) . Applying Lemma 7.8 for $H = G$ together with the exponential sum bound from Theorem 7.9 gives the desired result since without loss of generality we can assume that $\gamma < 1/2$. \square

The error bound in the above theorem is weaker than yet still comparable to the one from Theorem 7.6. Clearly, the dependence between β and $\gamma = \gamma(\beta)$ from Theorem 7.9 implies that γ degrades rapidly when β gets smaller. However, if we fix β we can still extract a constant fraction of bits by choosing, e.g., $N \approx p^{\gamma(\beta)/2}$ with the error bound ϵ'' negligible in $\log p$.

COMPUTING DISCRETE LOGARITHMS WITH GAUSSIAN PERIODS

As a by-product of our construction of the algorithm for computing a bijection $G \rightarrow \mathbb{Z}/M\mathbb{Z}$ for multiple arguments developed in [Chapter 6](#), we obtain a method for solving multiple instances of the DLP in $(\mathbb{Z}/p\mathbb{Z})^*$. We aim to amortize the cost for a single instance of the problem so the algorithm performs better than just applying the generic algorithm independently to each individual instance. Our basic idea is rather straightforward: extract the most time consuming step of the BSGS method, i.e., building a lookup table, run it only once, and reuse this result for every instance of the DLP. What is novel in our proposal is that the time and space complexities of this method are preserved when switching from the RAM model to a classical Turing machine. Clearly, both models are polynomially equivalent, but since we deal with exponential-level complexities the exact blow-up factor of such a reduction matters. [Diem \(2012\)](#) adapts the BSGS method to a multitape Turing machine keeping the sorting step of a lookup table. Our concluding solution does not require sorting at all, and it primarily addresses Turing machines with a single work-tape, where no subquadratic sorting algorithm exists ([Petersen 2008](#)).

First, we note that the space bound parameter s in [Theorem 6.10](#) can be in fact much larger, i.e., we can set it arbitrarily as long as $0 < s \leq P^+(M)$. This change requires the inner algorithm for computing discrete logarithms to be modified. We present the relevant adjustment of the BSGS method for online algorithm below.

Let $q \mid p - 1$ and $0 < s \leq q$, where s can be assumed to be an integer. Suppose we are given an instance of the DLP: for $a, b \in (\mathbb{Z}/p\mathbb{Z})^*$ with $\text{ord}(a) = q$ and $b \in \langle a \rangle$ find $0 \leq u < q$ such that $a^u = b$. Here and below, we focus on prime order sub-

groups $\langle a \rangle$, which is the hardest case of the DLP. A transition to general groups can be performed in the same standard way as it is done in the PH algorithm. Write $u = u_0 + u_1 \cdot s$ with unknowns u_0 and u_1 satisfying $0 \leq u_0 < s$ and $0 \leq u_1 \leq \lfloor q/s \rfloor$. We memoize values a^{-j} for $j = 0, \dots, s-1$ and store every pair (j, a^{-j}) in an array of s entries, say T , which is sorted (lexicographically) according to the second entry of each pair. Finally, for each $u_1 = 0, \dots, \lfloor q/s \rfloor$ we use a binary search to check whether $(a^s)^{u_1} \cdot b^{-1}$ appears in T . Finding a collision $a^{s \cdot u_1} \cdot b^{-1} = a^{-u_0}$ yields the desired solution of the DLP. The array T , once computed, can be reused to speed up subsequent calculations. That is, given another instance: $a_i, b_i \in (\mathbb{Z}/p\mathbb{Z})^*$ with $\text{ord}(a_i) = q$ and $b_i \in \langle a_i \rangle$ we compute $\log_{a_i} b_i$ as follows. We find u' and u'' such that $a^{u'} = a_i$ and $a^{u''} = b_i$ by searching in T like above but with b substituted with a_i and b_i , respectively. Then, $\log_{a_i} b_i = u''u'^{-1} \pmod{q}$. This way a single instance is solved in $O(\frac{q}{s}(\log s) \text{poly}(\log p)) = O(qs^{-1} \text{poly}(\log p))$ steps, and the pre-computation phase, where T is constructed, takes $O(q \text{poly}(\log p))$ time.

The use of a binary search in the aforesaid method implies that we work in the usual model of computations where each memory cell can be accessed in constant time. The fact that q and s are typically large makes the assumption about keeping whole T in the main memory unrealistic and justifies turning attention to external algorithms. Our motivation behind such an approach is similar to the one that historically led to introducing external sorting. We thus consider and analyze an algorithm for a classical Turing machine. In the solution we provide the array T remains unsorted, resides on a machine's tape, and can be scanned for a given value in linear time. As for complexity of the algorithm, we are unable to maintain the " $\frac{q}{s}$ -time vs. s -memory" trade-off, but we come close to. A downside of the construction is that the parameter s can no longer be chosen flexibly. Our method exploits invariance of Gaus-

sian periods, which have numerous applications. We briefly recall some relevant notions.

For a prime $q \mid p-1$ let $q-1 = s \cdot t$ be some known factorization of $q-1$. Our method works best when $q-1$ splits evenly, i.e., when $s \approx t \approx \sqrt{q}$, yet, in principle, s and t can be arbitrary. Set $\zeta_q := e^{2\pi i/q}$ to be a q th primitive complex root of unity. Consider the group $(\mathbb{Z}/q\mathbb{Z})^* = \langle h \rangle$ with some generator h and its subgroup $H := \langle h^s \rangle$. We define a *Gaussian period* η_j for $j = 0, \dots, s-1$ as the trace of $\zeta_q^{h^j}$ to the unique subfield of $\mathbb{Q}(\zeta_q)$ of degree s over \mathbb{Q} . That is,

$$\eta_j := \sum_{\alpha \in h^j H} \zeta_q^\alpha,$$

where the sum is taken over the coset $h^j H$ in $(\mathbb{Z}/q\mathbb{Z})^*$. Let $F = F_q \in \mathbb{Q}[X]$ be the minimal polynomial for η_0 , namely

$$F(X) := \prod_{j=0}^{s-1} (X - \eta_j).$$

We note that F is irreducible over \mathbb{Q} and has integer coefficients. When considered over a finite field, $\mathbb{Z}/p\mathbb{Z}$ in our case, using a canonical mapping $F \mapsto F \bmod p$ may possess a non-trivial factorization (and, in fact, for $q \mid p-1$ it does have such a factorization). In several salient applications, e.g., in the one by [Lenstra \(2002\)](#), it is required to keep F irreducible, which is typically ensured via Kummer's criterion. We, however, have to guarantee that much weaker property holds, specifically that F does not have repeated roots in $\mathbb{Z}/p\mathbb{Z}$. This is equivalent to non-vanishing of the discriminant Δ_F of F in $\mathbb{Z}/p\mathbb{Z}$.

THEOREM 8.1. *Suppose that p does not divide the discriminant Δ_F . There exists a deterministic Turing machine that given ℓ pairs (a_i, b_i) over $(\mathbb{Z}/p\mathbb{Z})^*$ for $i = 1, \dots, \ell$ with $\text{ord}(a_i) = q$ and $b_i \in \langle a_i \rangle$ computes ℓ discrete logarithms $\log_{a_i} b_i$ sequentially, and it does so within $O((q + \ell \max(s, t)) \text{poly}(\log p))$ total time using $O(s \log p)$ bits of additional space, i.e., the total memory excluding the space reserved for input and output data.*

PROOF. In our construction we assume that there are two tapes available on the machine. The first one plays the role of an input/output tape and is only used to store the initial configuration of (a_i, b_i) and resulting discrete logarithms. The other tape, which is limited to $O(s \log p)$ bits, functions as a “working” area to write intermediate data, including an array T described next.

First, the machine fixes a q th root of unity in $(\mathbb{Z}/p\mathbb{Z})^*$, say ε . Since each a_i is such a root we can, for instance, choose ε equal to a_1 . As we have already noted at the beginning of this chapter, it suffices to specify how to find $0 \leq u < q$ such that $\varepsilon^u = a$ for any given $a \in \langle \varepsilon \rangle$. Without loss of generality we can assume that $a \neq 1$, so $\text{ord}(a) = q$.

Define a polynomial $f \in (\mathbb{Z}/p\mathbb{Z})[Y]$ by

$$(8.2) \quad f(Y) := \sum_{\alpha \in H} Y^\alpha .$$

In the precomputation phase the machine calculates $f(\varepsilon^{h^j})$ and stores this value in the j th entry of the array T for $j = 0, \dots, s-1$. Filling T contributes $O(s \cdot t \text{poly}(\log p))$ to the time complexity if the naïve method of evaluating f is used, but this step is performed only once. Now, given a the machine determines $f(a)$ and scans T until an index j such that $f(a) = f(\varepsilon^{h^j})$ is found. Clearly, $u = \log_\varepsilon a$ belongs to $h^j H$ for some $0 \leq j < s$, so the value $f(a)$ is indeed present in the array. We claim that the index j is unique and T contains no duplicates. Suppose that $f(\varepsilon^{h^j}) = f(\varepsilon^{h^{j'}})$ for some $0 \leq j' < s$. Let $\Phi_q(Y) = 1 + Y + \dots + Y^{q-1}$ be the q th cyclotomic polynomial. Since $\mathbb{Z}(\zeta_q) \simeq \mathbb{Z}[Y]/(\Phi_q)$ there is a canonical projection modulo p , namely

$$\mathbb{Z}(\zeta_q)[X] \rightarrow (\mathbb{Z}[Y]/(p, \Phi_q))[X] \rightarrow (\mathbb{Z}[Y]/(p, Y - \varepsilon))[X] ,$$

that maps F to $F \bmod p = \prod_{j=0}^{s-1} (X - f(\varepsilon^{h^j}))$. The condition on the discriminant of F implies that $F \bmod p$ does not have repeated roots. Therefore, $f(\varepsilon^{h^j}) = f(\varepsilon^{h^{j'}})$ necessarily means that $j = j'$.

Now that j is found, the machine tests each $0 \leq j'' < t$ to find the one satisfying $a = \varepsilon^{h^j \cdot (h^s)^{j''}}$. The solution for the DLP instance is thus $u := h^{j+s j''}$. For each a , the complexity of calculating $f(a)$ and scanning T is $O((t+s) \text{poly}(\log p))$. Finding j'' incurs additional $O(t \text{poly}(\log p))$ time cost. \square

To compare this result with the BSGS method, we note that **Diem's** estimate of the running time of the latter implemented on a multitape machine is $O(q^{1/2}(\log q) \text{poly}(\log p))$, where the $\text{poly}(\log p)$ factor, arising from integer multiplication and exponentiation, is roughly of the same magnitude as ours. Therefore, if we look at the amortized cost for a single DLP instance in the case of balanced partitioning $s \approx t \approx q^{1/2}$ and $\ell = \Omega(q^{1/2})$, our algorithm performs on par with the generic method, even though the model we work in is less powerful. By **Petersen's** impossibility result, on single-tape machines the BSGS method does not offer any significant advantage over brute-force search.

As an immediate corollary of **Theorem 8.1**, under the same assumption on discriminant as above, there exists a bijective mapping $G_q \rightarrow \mathbb{Z}/q^{\alpha} \mathbb{Z}$ that can be computed by a deterministic Turing machine for a sequence of ℓ arguments in

$$O((q + \ell \max(s, t)) \text{poly}(\log p))$$

time and $O(s \log p)$ additional space. Clearly, this algorithm gives rise to calculating a bijective mapping $G \rightarrow \mathbb{Z}/M\mathbb{Z}$ provided that $p \nmid \Delta_{F_q}$ for all $q \mid M$. The last condition can be relaxed to some degree. Namely, if Q is a prime divisor of M such that $p \nmid \Delta_{F_Q}$, $Q-1 = s \cdot t$, and $q = O(\max(s, t))$ for all the remaining prime divisors q of M , then we can apply the algorithm from **Theorem 8.1** for Q -subgroups and calculate discrete logarithms for other factors q simply by exhaustive search. This would not increase the overall asymptotic complexity.

A "bottleneck" of the above algorithm is the step where the polynomial f is evaluated at some given point. This is indeed time

consuming as the formula (8.2) possibly consists of many terms. Regrettably, we were unable to come up with a more efficient way to do this other than the naïve method. An improvement in this matter could also speed up the entire algorithm, and therefore this problem looks like an interesting open question.

BIBLIOGRAPHY

- MANINDRA AGRAWAL, NEERAJ KAYAL & NITIN SAXENA (2002). PRIMES is in P. *Annals of Mathematics* **160**(2), 781–793.
- WILLIAM R. ALFORD, ANDREW GRANVILLE & CARL POMERANCE (1994). There are infinitely many Carmichael numbers. *Annals of Mathematics* **139**(3), 703–722. ISSN 0003–486X.
- NESMITH C. ANKENY (1952). The least quadratic non residue. *Annals of Mathematics* **55**, 65–72. ISSN 0003–486X.
- ERIC BACH (1997). Comments on search procedures for primitive roots. *Mathematics of Computation* **66**(220), 1719–1727.
- JEAN BOURGAIN (2005). More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory* **1**(1), 1–32.
- JEAN BOURGAIN, ALEXEY GLIBICHUK & SERGEI KONYAGIN (2006). Estimates for the number of sums and products and for exponential sums in fields of prime order. *Journal of the London Mathematical Society* **73**(2), 380–398.
- JEAN BOURGAIN, NETS KATZ & TERENCE TAO (2004). A sum-product estimate in finite fields, and applications. *Geometric & Functional Analysis GAFA* **14**(1), 27–57. ISSN 1016-443X. URL <http://dx.doi.org/10.1007/s00039-004-0451-1>.
- JEAN BOURGAIN & SERGEI KONYAGIN (2003). Estimates for the number of sums and products and for exponential sums over subgroups in fields of prime order. *Comptes Rendus Mathématique* **337**(2), 75 – 80. ISSN 1631-073X. URL <http://www.sciencedirect.com/science/article/pii/S1631073X03002814>.
- CÉLINE CHEVALIER, PIERRE-ALAIN FOUQUE, DAVID POINTCHEVAL & SÉBASTIEN ZIMMER (2009). Optimal randomness extraction from a Diffie-Hellman element. In *Advances in Cryptology - Proceedings of EUROCRYPT*

'09, ANTOINE JOUX, editor, volume 5479 of *Lecture Notes in Computer Science*, 572–589. Springer, Cologne, Germany.

OLIVIER CHEVASSUT, PIERRE-ALAIN FOUQUE, PIERRICK GAUDRY & DAVID POINTCHEVAL (2006). The twist-AUGmented technique for key exchange. In *PKC '06*, MOTI YUNG, YEVGENIY DODIS, AGGELOS KILIAS & TAL MALKIN, editors, volume 3958 of *Lecture Notes in Computer Science*, 410–426. Springer-Verlag.

BENNY CHOR & ODED GOLDREICH (1988). Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal of Computing* **17**(2), 230–261. ISSN 0097-5397.

AVIAD COHEN & AVI WIGDERSON (1989). Dispersers, deterministic amplification, and weak random sources (extended abstract). In *FOCS '89*, 14–19. IEEE Computer Society.

GIL COHEN, RAN RAZ & GIL SEGEV (2012). Non-malleable extractors with short seeds and applications to privacy amplification. In *IEEE Conference on Computational Complexity*, 298–308. IEEE. ISBN 978-1-4673-1663-7.

CLAUS DIEM (2012). On the complexity of some computational problems in the Turing model. Preprint.

YEVGENIY DODIS, XIN LI, TREVOR D. WOOLEY & DAVID ZUCKERMAN (2011). Privacy amplification and non-malleable extractors via character sums. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, *FOCS '11*, 668–677. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-4571-4.

YEVGENIY DODIS & ROBERTO OLIVEIRA (2003). On extracting private randomness over a public channel. In *RANDOM-APPROX*, SANJEEV ARORA, KLAUS JANSEN, JOSÉ D. P. ROLIM & AMIT SAHAI, editors, volume 2764 of *Lecture Notes in Computer Science*, 252–263. Springer. ISBN 3-540-40770-7.

YEVGENIY DODIS & DANIEL WICHS (2009). Non-malleable extractors and symmetric key cryptography from weak secrets. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, *STOC '09*, 601–610. ACM, New York, NY, USA. ISBN 978-1-60558-506-2.

KONRAD DURNOGA & BARTOSZ ŻRAŁEK (2013). On Randomness Extractors and Computing Discrete Logarithms in Bulk. Preprint (submitted to *Computational Complexity*).

AMOS FIAT & GERHARD J. WOEGINGER (editors) (1998). *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-64917-4.

PIERRE-ALAIN FOUQUE, DAVID POINTCHEVAL, JACQUES STERN & SÉBASTIEN ZIMMER (2006). Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes. In *Automata, Languages and Programming*, MICHELE BUGLIESI, BART PRENEEL, VLADIMIRO SASSONE & INGO WEGENER, editors, volume 4052 of *Lecture Notes in Computer Science*, 240–251. Springer Berlin Heidelberg. ISBN 978-3-540-35907-4.

SHUHONG GAO (1997). Elements of provable high orders in finite fields. *Proceedings of the American Mathematical Society* **127**, 1615–1623.

JOACHIM VON ZUR GATHEN & IGOR E. SHPARLINSKI (2001). Gauß Periods in Finite Fields. In *Finite Fields and Applications*, DIETER JUNGnickel & HARALD NIEDERREITER, editors, 162–177. Springer-Verlag. ISBN 3-540-41109-7.

JOACHIM VON ZUR GATHEN & IGOR E. SHPARLINSKI (2013). Generating safe primes. Preprint.

ANDREW GRANVILLE & CARL POMERANCE (1990). On the least prime in certain arithmetic progressions. *Journal of the London Mathematical Society* **2**(2), 193–200.

THOMAS HOLENSTEIN (2006). Pseudorandom generators from one-way functions: A simple construction for any hardness. In *In 3rd Theory of Cryptography Conference – (TCC '06)*, SHAI HALEVI & TAL RABIN, editors, *Lecture Notes in Computer Science*. Springer-Verlag.

RUSSELL IMPAGLIAZZO, LEONID A. LEVIN & MICHAEL LUBY (1989). Pseudo-random generation from one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing, STOC '89*, 12–24. ACM, New York, NY, USA. ISBN 0-89791-307-8.

HENRYK IWANIEC & EMMANUEL KOWALSKI (2004). *Analytic Number Theory*. Number vol. 53 in American Mathematical Society Colloquium Publications. American Mathematical Society, Providence, Rhode Island, USA. ISBN 0-8218-3633-1.

SERGEI KONYAGIN & CARL POMERANCE (1996). On primes recognizable in deterministic polynomial time. In *The Mathematics of Paul Erdős*, volume 13 of *Algorithms Combin.*, 176–198. Springer, Berlin.

HENDRIK W. LENSTRA (2002). Primality Testing with Gaussian Periods. In *FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science, 22nd Conference Kanpur, India, December 12-14, 2002, Proceedings*, MANINDRA AGRAWAL & ANIL SETH, editors, volume 2556 of *Lecture Notes in Computer Science*, 1. Springer. ISBN 3-540-00225-1.

XIN LI (2012a). Non-Malleable Condensers for Arbitrary Min-Entropy, and Almost Optimal Protocols for Privacy Amplification.

XIN LI (2012b). Non-malleable extractors, two-source extractors and privacy amplification. In *FOCS '12*, 688–697. IEEE Computer Society, Los Alamitos, CA, USA. ISSN 0272-5428.

NOAM NISAN & DAVID ZUCKERMAN (1993). More deterministic simulation in logspace. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, 235–244. ACM, New York, NY, USA. ISBN 0-89791-591-7.

HOLGER PETERSEN (2008). Sorting and element distinctness on one-way Turing machines. In *LATA*, CARLOS MARTÍN-VIDE, FRIEDRICH OTTO & HENNING FERNAU, editors, volume 5196 of *Lecture Notes in Computer Science*, 433–439. Springer. ISBN 978-3-540-88281-7.

JONATHAN PILA (1990). Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation* 55(192), 745–763. ISSN 00255718.

STEPHEN POHLIG & MARTIN HELLMAN (1978). An improved algorithm for computing logarithms over $\text{GF}(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory* 24(1), 106–110. ISSN 0018-9448.

CARL POMERANCE (2002). The expected number of random elements to generate a finite abelian group. *Periodica Mathematica Hungarica* 43(1-2), 191–198.

CARL POMERANCE & IGOR SHPARLINSKI (2002). Smooth orders and cryptographic applications. In *ANTS*, CLAUS FIEKER & DAVID R. KOHEL, editors, volume 2369 of *Lecture Notes in Computer Science*, 338–348. Springer. ISBN 3-540-43863-7.

ANUP RAO (2007). An exposition of Bourgain's 2-source extractor. *Electronic Colloquium on Computational Complexity* **14**. Technical Report TR07-034.

WOLFGANG M. SCHMIDT (1976). *Equations Over Finite Fields: An Elementary Approach*. Number no. 536 in Lecture Notes in Mathematics. Springer-Verlag. ISBN 9783540078555.

RONEN SHALTIEL (2002). Recent Developments in Explicit Constructions of Extractors. *Bulletin of the EATCS* **77**, 67–95.

DANIEL SHANKS (1971). Class number, a theory of factorization, and genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, 415–440. Amer. Math. Soc., Providence, Rhode Island, USA.

VICTOR SHOUP (1990). Searching for primitive roots in finite fields. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing, STOC '90*, 546–554. ACM, New York, NY, USA. ISBN 0-89791-361-2.

ANDREAS STEIN & EDLYN TESKE (2005). Optimized baby step-giant step methods. *Journal of the Ramanujan Mathematical Society* **20**(1), 1–32.

DAVID TERR (2000). A modification of Shanks' baby-step giant-step algorithm. *Mathematics of Computation of the American Mathematical Society* **69**(230), 767–773.

LUCA TREVISAN (1999). Construction of extractors using pseudo-random generators (extended abstract). In *STOC '99*, 141–148. Atlanta, Georgia, USA.

SEBASTIAN WEDENIWSKI (2001). *Primality Tests on Commutator Curves*. Ph.D. thesis, Eberhard-Karls-Universität Tübingen.

TRIANAFYLLOS XYLOURIS (2011). *Über die Nullstellen der Dirichletschen L-Funktionen und die kleinste Primzahl in einer arithmetischen Progression*. Ph.D. thesis, Mathematisch-Naturwissenschaftliche Fakultät der Universität Bonn.

DAVID ZUCKERMAN (1990). General weak random sources. In *FOCS '90*, 534–543. IEEE Computer Society, Los Alamitos, CA, USA.