Electronic Colloquium on Computational Complexity, Report No. 147 (2024)

Pseudo-Deterministic Construction of Irreducible Polynomials over Finite Fields

Shanthanu S Rai *

Abstract

We present a polynomial-time pseudo-deterministic algorithm for constructing irreducible polynomial of degree d over finite field $\mathbb{F}_q$. A pseudo-deterministic algorithm is allowed to use randomness, but with high probability it must output a canonical irreducible polynomial. Our construction runs in time $\tilde{O}(d^4 \log^4 q)$.

Our construction extends Shoup's deterministic algorithm (FOCS 1988) for the same problem, which runs in time $\tilde{O}(d^4 p^{\frac{1}{2}} \log^4 q)$ (where p is the characteristic of the field $\mathbb{F}_q$). Shoup had shown a reduction from constructing irreducible polynomials to factoring polynomials over finite fields. We show that by using a fast randomized factoring algorithm, the above reduction yields an efficient pseudo-deterministic algorithm for constructing irreducible polynomials over finite fields.

1 Introduction

A polynomial $f(X)$ over a finite field $\mathbb{F}_q$ (q is a prime power) is said to be irreducible if it doesn't factor as $f(X) = g(X)h(X)$ for some non-trivial polynomials $g(X)$ and $h(X)$. Irreducible polynomials over finite fields are algebraic analogues of primes numbers over integers. It is natural to ask if one can construct an irreducible polynomial of degree d over $\mathbb{F}_q$ efficiently. Constructing these irreducible polynomials are important since they yield explicit construction of finite fields of non-prime order. Working over such non-prime finite fields is crucial in coding theory, cryptography, pseudo-randomness and derandomization. Any algorithm that constructs irreducible polynomials of degree d over $\mathbb{F}_q$ would output $d \log q$ bits, so we expect an efficient algorithm for constructing irreducible polynomials would run in time $\text{poly}(d, \log q)$.

About $\frac{1}{d}$ fraction of polynomials of degree d are irreducible over $\mathbb{F}_q$ [9, Ex. 3.26 and 3.27]. This gives a simple "trial and error" randomized algorithm for constructing irreducible polynomials, namely, pick a random degree d polynomial and check if it is irreducible. We can use Rabin's algorithm [10] for checking if a polynomial is irreducible, which can be implemented in $\tilde{O}(d \log^2 q)$ [7, Section 8.2][1]. In order to improve the probability of finding an irreducible polynomial to $\frac{1}{2}$, we sample about d polynomials of degree d and check if any one of them is irreducible. Thus, the "trial and error" algorithm runs in time $\tilde{O}(d^2 \log^2 q)$. Couveignes and Lercier [5] give an alternative randomized algorithm that runs in time $\tilde{O}(d \log^5 q)$, which is optimal in the exponent of d. Their algorithm constructs irreducible polynomials by using isogenies between elliptic curves.

*Tata Institute of Fundamental Research, Mumbai, India. Email: shanthanu.rai@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500.

[1] $\tilde{O}$ notation omits log factors in d and $\log q$

ISSN 1433-8092

Motivated by this, it is natural to ask if there is also an efficient deterministic algorithm for constructing irreducible polynomials. In the 80s, some progress was made towards this problem. Adleman and Lenstra [1] gave an efficient deterministic algorithm for this problem conditional on the generalized Riemann hypotheses. They also gave an unconditional deterministic algorithm which outputs an irreducible polynomial of degree approximately d. Shoup [12] gives a deterministic algorithm of constructing degree d irreducible polynomial which runs in time $\tilde{O}(d^4 p^{\frac{1}{2}} \log^4 q)$ (where p is the characteristic of $\mathbb{F}_q$). So, Shoup's algorithm is efficient for fields of small characteristic ($p << d$). But when p is large (say super exponential in d), the algorithm does not run in polynomial time due to the $p^{\frac{1}{2}}$ factor in the run time. Since then, there hasn't been much progress towards this problem and in particular, the problem of efficient and unconditional deterministic construction of irreducible polynomials over $\mathbb{F}_q$ remains open! In fact, the special case of efficient and unconditional deterministic construction of quadratic non-residues in $\mathbb{F}_p$ is also open.

One can ask similar questions in the integer world, namely, "How to efficiently construct n-bit prime numbers?". By the Prime Number Theorem, there are about $\frac{1}{n}$ n-bit prime numbers (note the similarity between density of primes and density of irreducible polynomials over $\mathbb{F}_q$). Again this gives a simple randomized algorithm of just sampling a random n-bit number and checking if it's prime using AKS primality test [2]. But here too, there is no known efficient deterministic algorithm for constructing n-bit prime numbers [13].

Due to the difficulty in finding deterministic algorithms for these problems, we ask a slightly weaker but related question. Are there efficient pseudo-deterministic algorithms for these problems?

Definition 1.1. *A pseudo-deterministic algorithm is a randomized algorithm which for a given input, generates a canonical output with probability at least $\frac{1}{2}$.* $\diamondsuit$

Gat and Goldwasser [6] first introduced the notion of pseudo-deterministic algorithm (they had called it Bellagio algorithm). Pseudo-deterministic algorithm can be viewed as a middle ground between a randomized and a deterministic algorithm. From an outsider's perspective, a pseudo-deterministic algorithm seems like a deterministic algorithm in the sense that with high probability it outputs the same output for a given input. The breakthrough result of Chen et al. [4] gave a polynomial-time pseudo-deterministic algorithm for constructing n-bit prime numbers in the infinitely often regime.

Theorem 1.2. *There is a randomized polynomial-time algorithm B such that, for infinitely many values of n, $B(1^n)$ outputs a canonical n-bit prime p_n with high probability.*

In particular, their algorithm doesn't give valid outputs for all values of the input n. Surprisingly, their algorithm is based on complexity theoretic ideas, and not number theoretic ideas. In fact, they show a more general result that if a set of strings Q are "dense" and it is "easy" to check if a string x is in Q, then there is an efficient pseudo-deterministic algorithm for generating elements of Q of a particular length in the infinitely often regime. Both prime numbers and irreducible polynomials over $\mathbb{F}_q$ satisfy this property. Thus, this gives an efficient pseudo-deterministic algorithm for constructing irreducible polynomials over $\mathbb{F}_q$ in the infinitely often regime.

But not only does this algorithm not work for all d, there are no good density bounds for the fraction of d where the algorithm gives valid output. So it is natural to ask if we can extend this result to all values of degree d over all finite fields $\mathbb{F}_q$. In this paper, we present a more direct pseudo-deterministic algorithm for constructing irreducible polynomials over $\mathbb{F}_q$ (for all degrees d) which crucially relies on the structure of irreducible polynomials. Our result extends Shoup's [12] deterministic algorithm for constructing irreducible polynomials. Shoup reduces the problem of constructing irreducible polynomials to factoring polynomials over $\mathbb{F}_q$. We observe that by making use of the fast randomized factoring algorithm, and the "canonization" process described by Gat and Goldwasser [6] for computing q-th residues over $\mathbb{F}_p$, the above reduction yields an efficient pseudo-deterministic algorithm for constructing irreducible polynomials over $\mathbb{F}_q$.

Theorem 1.3. *There is a pseudo-deterministic algorithm for constructing an irreducible polynomial of degree d over $\mathbb{F}_q$ (q is prime power) in expected time $\tilde{O}(d^4 \log^4 q)$.*

2 Overview

As mentioned earlier, Shoup's deterministic algorithm [12] is efficient for fields of small characteristic. We extend Shoup's algorithm and make it efficient over all fields, but at the cost of making the algorithm pseudo-deterministic. In order to see the main ideas involved, let's consider a toy problem of constructing irreducible polynomial of degree 2 over $\mathbb{F}_p$ (p is prime). Suppose we could get our hands on some quadratic non residue α, then $X^2 - \alpha$ would be irreducible. There are $\frac{p-1}{2}$ quadratic non residues in $\mathbb{F}_p$, so if we randomly pick an $\alpha \in \mathbb{F}_p$ and output $X^2 - \alpha$, it would be irreducible with about $\frac{1}{2}$ probability. But this approach wouldn't be pseudo-deterministic, since in each run we will very likely choose different α.

In order to obtain a canonical quadratic non residue α, we first set $\alpha = -1$ and repeatedly perform $\alpha \leftarrow \sqrt{\alpha}$ (choosing the smallest square root) until α is a quadratic non residue. Here, β is a square root of α if $\beta^2 = \alpha \pmod p$. For computing the square root, we can use Cantor-Zassenhaus randomized factoring algorithm [3]. In Example 2.1, we illustrate the above strategy over a specific finite field. Algorithm 1 implements this strategy.

Example 2.1. *Let's try to pseudo-deterministically construct a quadratic non-residue in $\mathbb{F}_{73}$. We first set $\alpha = -1$. Square roots of $-1 \pmod{73}$ are 27 and 46. The square roots are computed using Cantor-Zassenhaus randomized factoring algorithm [3].*

We choose the smallest square root 27 and set $\alpha = 27$. Square roots of $27 \pmod{73}$ are 10 and 63. We choose the smallest square root 10 and set $\alpha = 10$. Since 10 is a quadratic non-residue, we output 10 (we use Euler's criterion[2] to check if 10 is a quadratic non-residue). $\diamondsuit$

Algorithm 1 Pseudo-deterministically constructing irreducible polynomial of degree 2 over $\mathbb{F}_p$

$\alpha \leftarrow -1$
while α is a quadratic residue **do**
    Factorize $X^2 - \alpha = (X - \beta_1)(X - \beta_2)$
    $\alpha \leftarrow \min(\beta_1, \beta_2)$
end while
Output $X^2 - \alpha$

Suppose $p - 1 = 2^k l$ (where l is odd). Each time we take square root, the order[3] of $\alpha \pmod p$ doubles. Since the order of α divides $|\mathbb{F}_p^*| = 2^k l$ (by Lagrange's theorem), we can repeatedly take square roots in Algorithm 1 at most k times. Thus, Algorithm 1 will terminate with at most $\log p$ iterations of the while loop. This algorithm is based on Gat and Goldwasser's algorithm [6] for

[2]Euler's criterion: For odd prime p, a is a quadratic non residue iff $a^{(p-1)/2} = -1$

[3]Order of α is the least integer $k > 0$ such that $\alpha^k = 1$ in $\mathbb{F}_p$

computing q-th residues over $\mathbb{F}_p$. The algorithm is pseudo-deterministic since at each iteration of the while loop, we "canonize" our choice of square root by picking the smallest one among the two choices. Note that we used Euler's criterion for checking if α is a quadratic residue or not in Line 2.

We can generalize the above ideas for constructing irreducible polynomials over finite fields. Shoup [12] showed that constructing irreducible polynomials over $\mathbb{F}_p$ reduces to finding q-th non residues over appropriate field extensions (q is prime). These q-th non residues can be pseudo-deterministically constructed using similar techniques as in Algorithm 1.

The rest of the paper is organized as follows. We start with some preliminaries in Section 3. In Section 4, we will reduce the problem of constructing irreducible polynomials over extensions fields $\mathbb{F}_{p^k}$ to constructing them over $\mathbb{F}_p$. Section 5 will make use of Shoup's observation mentioned in previous paragraph to construct irreducible polynomials over $\mathbb{F}_p$. Finally, in Section 6 we conclude with some open problems.

3 Preliminaries

3.1 Pseudo-deterministic algorithms

We defined pseudo-deterministic algorithm to be randomized algorithm which for a given input, generates a canonical output with probability at least $\frac{1}{2}$. In this paper, whenever the pseudo-deterministic algorithm doesn't generate a canonical output, it just fails and doesn't give any valid output. In such cases, we can just rerun the algorithm until we get some valid output (which is bound to be canonical). Now the runtime of the algorithm will be random, but the expected runtime will be (asymptotically) same as the original runtime.

For all the pseudo-deterministic algorithms in this paper, we report the expected run time in the above sense. These algorithms always generate a canonical output, but the amount of time they take to do so is random.

3.2 Finite Field primer

In this subsection, we go over some basic facts about finite fields that will be useful in later sections.

3.2.1 Splitting field

A polynomial $h(X) \in \mathbb{K}[X]$ may not factorize fully into linear factors over the field $\mathbb{K}$. Suppose $\mathbb{F}$ is the smallest extension of $\mathbb{K}$ such that $h(X)$ fully factorizes into linear factors over $\mathbb{F}$. In other words, there exists $\alpha_1, \alpha_2, \ldots \alpha_k \in \mathbb{F}$ such that,

$$h(X) = (X - \alpha_1)(X - \alpha_2) \cdots (X - \alpha_k)$$

Then $\mathbb{F}$ is the called the splitting field of $h(X)$ over $\mathbb{K}$ [9, Definition 1.90]. Note that for any other extension of $\mathbb{K}$ that is a proper subfield of $\mathbb{F}$, $h(X)$ will not fully factorize into linear factors.

3.2.2 Structure of Finite Fields

For every prime power p^n (p is prime), there exists a finite field of size p^n and all finite fields of size p^n are isomorphic to each other [9, Theorem 2.5].

Theorem 3.1 (Existence and Uniqueness of Finite Fields). *For every prime p and every positive integer n there exists a finite field with p^n elements. Any finite field with $q = p^n$ elements is isomorphic to the splitting field of $X^q - X$ over $\mathbb{F}_p$.*

Thus, elements of $\mathbb{F}_{p^n}$ are roots of $X^{p^n} - X$. From this, we get the following generalization of Fermat's little theorem for finite fields:

Theorem 3.2 (Fermat's little theorm for finite fields). *If $\alpha \in \mathbb{F}_{p^n}$, then $\alpha^{p^n} = \alpha$. Conversely, if α is in some finite field and $\alpha^{p^n} = \alpha$, then $\alpha \in \mathbb{F}_{p^n}$.*

The below theorem gives the necessary and sufficient condition for a finite field $\mathbb{F}_{p^m}$ to be a subfield of another finite field $\mathbb{F}_{p^n}$ [9, Theorem 2.6].

Theorem 3.3 (Subfield Criterion). *Let $\mathbb{F}_q$ be a finite field with $q = p^n$ elements. Then every subfield of $\mathbb{F}_q$ has order p^m, where m is the positive divisor of n. Conversely, if m is the positive divisor of n, then there is exactly one subfield of $\mathbb{F}_q$ with p^m elements.*

From Theorem 3.2 and Theorem 3.3, we get the following useful lemma:

Lemma 3.4. *Suppose α is some finite field element. Let k be the smallest integer greater than 0 such that $\alpha^{p^k} = \alpha$. Then, $\mathbb{F}_{p^k}$ is the smallest extension of $\mathbb{F}_p$ that contains α. In other words, $\alpha \in \mathbb{F}_{p^k}$ and for all $1 \leq k' < k$, $\alpha \notin \mathbb{F}_{p^{k'}}$.*

3.2.3 Conjugates and Minimal polynomial

Let $f(X)$ be an irreducible polynomial of degree n over $\mathbb{F}_q$ (q is prime power). Then, $f(X)$ has some root $\alpha \in \mathbb{F}_{q^n}$. Also, the elements $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}$ are all distinct and are the roots of $f(X)$ [9, Theorem 2.14].

$$f(X) = (X - \alpha)(X - \alpha^q)(X - \alpha^{q^2}) \cdots (X - \alpha^{q^{n-1}})$$

The splitting field of $f(X)$ with respect to $\mathbb{F}_q$ is $\mathbb{F}_{q^n}$ [9, Corollary 2.15]. The minimal polynomial of α over $\mathbb{F}_q$ is $f(X)$.

Above, the roots of $f(X)$ are all of the form α^{q^i}. We will call such elements conjugates α with respect to $\mathbb{F}_q$:

Definition 3.5. *Let $\mathbb{F}_{q^n}$ be an extension of $\mathbb{F}_q$ and let $\beta \in \mathbb{F}_{q^n}$. Then, $\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{n-1}}$ are called the conjugates of β with respect to $\mathbb{F}_q$.* ♢

In the following sections, we will be using the below lemma to show that certain polynomials are irreducible.

Lemma 3.6 (Minimal polynomial of $\beta \in \mathbb{F}_{q^n}$). *Suppose $\beta \in \mathbb{F}_{q^n}$ and conjugates of β with respect to $\mathbb{F}_q$ are all distinct. Then the minimal polynomial of β over $\mathbb{F}_q$ has degree n and is of the form:*

$$g(X) = (X - \beta)(X - \beta^q)(X - \beta^{q^2}) \cdots (X - \beta^{q^{n-1}})$$

Thus, $g(X) \in \mathbb{F}_q[X]$ is an irreducible polynomial.

Proof. The minimal polynomial $g(X)$ of β over $\mathbb{F}_q$ is the smallest degree polynomial in $\mathbb{F}_q[X]$ such that $g(\beta) = 0$. Since, $g(\beta^{q^i}) = g(\beta)^{q^i} = 0$, all conjugates of β are roots of $g(X)$. Hence, degree of $g(X)$ is at least n (since the conjugates are all distinct). Also since $\beta \in \mathbb{F}_{q^n}$, degree of $g(X)$ is at most n. Thus, the degree of $g(X)$ is n.

Thus, $g(X) = (X - \beta)(X - \beta^q)(X - \beta^{q^2}) \cdots (X - \beta^{q^{n-1}})$. Since $g(X)$ is a minimal polynomial of β over $\mathbb{F}_q$, it will be in $\mathbb{F}_q[X]$ and is irreducible. □

3.2.4 Representing finite field elements

Throughout the paper, we assume that extension fields $\mathbb{F}_{p^k}$ are given to us as $\mathbb{F}_p[X]/(f(X))$, where $f(X)$ is an irreducible polynomial of degree k over $\mathbb{F}_p$ (refer [8] for working with other representations). Each element in $\mathbb{F}_p[X]/(f(X))$ can be viewed as a polynomial with degree at most k over $\mathbb{F}_p$. The coefficient vectors of these polynomials are in $\mathbb{F}_p^k$. This gives a natural isomorphism $\Phi : \mathbb{F}_{p^k} \to \mathbb{F}_p^k$. In $\mathbb{F}_p^k$, we can order elements in lexicographic order in the natural sense.

Definition 3.7. *We say that $\alpha \in \mathbb{F}_{p^k}$ is lexicographically smaller than $\beta \in \mathbb{F}_{p^k}$, if $\Phi(\alpha) \in \mathbb{F}_p^k$ is lexicographically smaller than $\Phi(\beta) \in \mathbb{F}_p^k$.* ♢

In the above definition, we compare the coordinates of $\Phi(\alpha)$ and $\Phi(\beta)$ by fixing some ordering on elements on $\mathbb{F}_p$ (for e.g., we can consider the natural ordering one gets from the additive group structure of $\mathbb{F}_p$). Checking if $\Phi(\alpha)$ is lexicographically smaller than $\Phi(\beta)$ requires k comparisons, with each comparison taking $O(\log p)$ time. Thus, overall it takes $O(k \log p)$ time to check if $\Phi(\alpha)$ is lexicographically smaller than $\Phi(\beta)$.

Similarly, we can define a lexicographic ordering on polynomials over $\mathbb{F}_{p^k}$.

Definition 3.8. *Suppose we are given two polynomials $g(X)$ and $h(X)$ of degree d over $\mathbb{F}_{p^k}$. Then we say that $g(X)$ is lexicographically smaller than $h(X)$ if the coefficient vector of $g(X)$ is lexicographically smaller than coefficient vector of $h(X)$ (the coefficients are compared using Φ).* ♢

Checking if $g(X)$ is lexicographically smaller than $h(X)$ requires $d + 1$ comparisons, with each comparison taking $O(k \log p)$ time. Thus, overall it takes $O(dk \log p)$ time to check if $g(X)$ is lexicographically smaller than $h(X)$.

Lemma 3.9 (Picking lexicographically smallest polynomial)**.** *Suppose we are given n polynomials $f_1(X), f_2(X), \ldots, f_n(X)$ of degree d over $\mathbb{F}_{p^k}$. Then, there is an algorithm that outputs the lexicographically smallest polynomial among them in $O(ndk \log p)$ time.*

Proof. We go over each polynomial $f_i(X)$ one by one, checking if $f_i(X)$ is lexicographically smaller than the lexicographically smallest polynomial we have seen so far. Since each comparison takes $O(dk \log p)$ time, and we do at most n comparisons, the algorithm runs in $O(ndk \log p)$ time. □

3.3 Equal degree polynomial factorization

Shoup [12] reduced constructing irreducible polynomials to factoring polynomials over finite field. It turns out that the reduction factors polynomials whose irreducible factors all have same degree. Hence, equal degree factorization is a crucial sub-routine for constructing irreducible polynomials. There are several fast randomized equal degree factorization algorithms, and below we mention one of them:

Theorem 3.10 (Equal degree factorization)**.** *Suppose $f(X)$ is a polynomial of degree d over $\mathbb{F}_q$ (q is prime power) which factors into irreducible polynomials of equal degree. Then, the equal degree factorization algorithm by von zur Gathen & Shoup [15] factors $f(X)$ in expected time $\tilde{O}(d \log^2 q)$.*

4 Construction of irreducible polynomials over extension fields $\mathbb{F}_{p^k}$

We first show in Algorithm 2 that constructing irreducible polynomials over extension fields $\mathbb{F}_{p^k}$ can be reduced to constructing irreducible polynomials over $\mathbb{F}_p$ (p is prime). Theorem 4.1 shows the correctness and running time of Algorithm 2.

Algorithm 2 Pseudo-deterministic construction of irreducible polynomials over $\mathbb{F}_{p^k}$

Input: Degree d
Output: Irreducible polynomial of degree d over $\mathbb{F}_{p^k}$

1: Pseudo-deterministically construct irreducible polynomial $f(X)$ over $\mathbb{F}_p$ of degree dk.
2: Factor $f(X) = \prod_{i=0}^{k-1} f_i(X)$ over $\mathbb{F}_{p^k}$ using Theorem 3.10.
3: Output the lexicographically smallest factor $f_i(X)$.

Theorem 4.1 (Correctness and Running time of Algorithm 2)**.** *Suppose there is a pseudo-deterministic algorithm for constructing irreducible polynomials of degree l over $\mathbb{F}_p$ (p prime), that runs in expected time $T(l,p)$. Then Algorithm 2 pseudo-deterministically constructs irreducible polynomials of degree d over extension field $\mathbb{F}_{p^k}$ in expected time $T(dk,p) + \tilde{O}(dk^3 \log p)$.*

Proof. Algorithm 2 first constructs an irreducible polynomial $f(X)$ of degree dk over F_p. Note that $\mathbb{F}_p[X]/(f(X))$ is isomorphic to $\mathbb{F}_{p^{dk}}$. Some $\alpha \in \mathbb{F}_{p^{dk}}$ will be a root of $f(X)$. The conjugates of α with respect to $\mathbb{F}_p$ are all distinct and are the roots of $f(X)$ (refer Section 3.2.3):

$$f(X) = (X-\alpha)(X-\alpha^p)(X-\alpha^{p^2})\cdots(X-\alpha^{p^{dk-2}})(X-\alpha^{p^{dk-1}})$$

Rearranging the above terms, we get:

$$\begin{aligned}
f(X) =& \Big[(X-\alpha)(X-\alpha^{p^k})(X-\alpha^{p^{2k}})\cdots(X-\alpha^{p^{(d-1)k}})\Big] \\
& \Big[(X-\alpha^p)(X-\alpha^{p^{k+1}})(X-\alpha^{p^{2k+1}})\cdots(X-\alpha^{p^{(d-1)k+1}})\Big] \\
& \Big[(X-\alpha^{p^2})(X-\alpha^{p^{k+2}})(X-\alpha^{p^{2k+2}})\cdots(X-\alpha^{p^{(d-1)k+2}})\Big] \\
& \vdots \\
& \Big[(X-\alpha^{p^{(k-1)}})(X-\alpha^{p^{k+(k-1)}})(X-\alpha^{p^{2k+(k-1)}})\cdots(X-\alpha^{p^{(d-1)k+(k-1)}})\Big] \\
=& \prod_{i=0}^{k-1}\prod_{j=0}^{d-1}(X-\alpha^{p^{jk+i}}) \\
\coloneqq& \prod_{i=0}^{k-1} f_i(X)
\end{aligned}$$

Let $q = p^k$. $f_i(X)$ has degree d and its roots are conjugates of $\alpha^{p^i} \in \mathbb{F}_{q^d}$ with respect to $\mathbb{F}_q$ (which are all distinct). Thus, from Lemma 3.6, $f_i(X) \in \mathbb{F}_q[X]$ is the minimal polynomial of α^{p^i} over $\mathbb{F}_q$, and hence $f_i(X)$ is irreducible over $\mathbb{F}_q$. So, we can use Theorem 3.10 to factorize $f(X)$ over $\mathbb{F}_q$, obtaining all factors $f_i(X)$ of degree d. We then use Lemma 3.9 to output the lexicographically smallest factor among $f_i(X)$. Let the lexicographically smallest factor be denoted by $f_{i^*}(X)$. Given a polynomial $f(X)$ of degree dk, $f_{i^*}(X)$ is canonical. Thus, the above construction is pseudo-deterministic.

For the running time, it takes $T(dk, p)$ time to construct $f(X)$, and then $\tilde{O}(dk^3 \log p)$ time to factor $f(X)$ over field $\mathbb{F}_{p^k}$ (from Theorem 3.10). Finally, choosing $f_{i^*}(X)$ among $f_i(X)$ can be computed in time $O(dk^2 \log p)$ (from Lemma 3.9). Thus, the overall running time of the algorithm is $T(dk, p) + \tilde{O}(dk^3 \log p)$. □

5 Construction of irreducible polynomials over $\mathbb{F}_p$

Shoup's algorithm reduces constructing irreducible polynomials over $\mathbb{F}_p$ to finding q-th non residues in splitting field of $X^q - 1$, for all prime divisors q of d (and $q \neq p$). For completeness, we reproduce the theorem below and refer to Theorem 2.1 in [12] for it's proof.

Theorem 5.1 (Reduction to finding q-th non residues)**.** *Assume that for each prime $q \mid d$, $q \neq p$, we are given a splitting field $\mathbb{K}$ of $X^q - 1$ over $\mathbb{F}_p$ and a q-th non residue in $\mathbb{K}$. Then we can find an irreducible polynomial over $\mathbb{F}_p$ of degree d deterministically with $\tilde{O}(d^4 \log p + \log^2 p)$ operations in $\mathbb{F}_p$.*

Shoup constructs the splitting field[4] $\mathbb{K}$ of $X^q - 1$ over $\mathbb{F}_p$ and a q-th non residue in $\mathbb{K}$ by reducing to deterministic polynomial factorization. Since no known efficient deterministic factoring algorithms are known, his algorithm is not efficient for finite fields of large characteristic. In this section, we will find a canonical splitting field $\mathbb{K}$ and a canonical q-th non residues by using a fast randomized factoring algorithm. Thus, we obtain an efficient algorithm for constructing irreducible polynomials over $\mathbb{F}_p$, but at the cost of making the algorithm pseudo-deterministic.

For each prime $q \mid d, q \neq p$, we will pseudo-deterministically construct a splitting field $\mathbb{K}$ of $X^q - 1$ over $\mathbb{F}_p$ and find a q-th non residue in $\mathbb{K}$. To this end, we first analyze the factorization of $X^q - 1 \in \mathbb{F}_p[X]$.

Lemma 5.2. *Consider the polynomial $X^q - 1 \in \mathbb{F}_p[X]$ (p, q are prime numbers). Let k be the smallest integer greater than 0 such that $q \mid p^k - 1$ (in other words, k is the order of $p \pmod q$). Then,*

1. *The splitting field of $X^q - 1$ over $\mathbb{F}_p$ is $\mathbb{F}_{p^k}$*
2. $X^q - 1 = (X - 1)g_1(X)g_2(X) \cdots g_{\frac{q-1}{k}}(X)$

   where $g_i(X) \in \mathbb{F}_p[X]$ are irreducible polynomials of degree k.

Proof. Let $\mathbb{K}$ be the splitting field of $X^q - 1$ over $\mathbb{F}_p$. The roots of $X^q - 1$ in $\mathbb{K}$ are by definition the q-th roots of unity. Suppose $\omega \in \mathbb{K}$ is some primitive q-th root of unity. Then, $\{1, \omega, \omega^2, \ldots, \omega^{q-1}\}$

[4]Shoup constructs an irreducible polynomial $g(X) \in \mathbb{F}_p[X]$ such that $\mathbb{F}_p/(g(X))$ is isomorphic to splitting field of $X^q - 1$ over $\mathbb{F}_p$

are all the q-th roots of unity, and they form a multiplicative subgroup in $\mathbb{K}^*$. In fact, since q is prime, each of $\{\omega, \omega^2, \ldots, \omega^{q-1}\}$ is a primitive q-th root of unity.

Since $\omega^q = 1$, we have $\omega^{p^k} = \omega$ and hence from Theorem 3.2, $\omega \in \mathbb{F}_{p^k}$. By definition of k, k is the smallest integer greater than 0 such that $\omega^{p^k} = \omega$. So from Lemma 3.4, $\mathbb{F}_{p^k}$ is the smallest extension of $\mathbb{F}_p$ that contains ω. $X^q - 1$ splits linearly as:

$$X^q - 1 = (X-1)(X-\omega)(X-\omega^2)\cdots(X-\omega^{q-1})$$

$\mathbb{F}_{p^k}$ is the smallest extension of $\mathbb{F}_p$ that contains all the roots of X^q-1. Thus, $\mathbb{F}_{p^k}$ is the splitting field of $X^q - 1$. We next consider the factorization pattern of $X^q - 1$ over $\mathbb{F}_p$.

Let G be the multiplicative group of integers modulo q. Since q is prime, elements of G are $\{1, 2, \ldots, q-1\}$. Consider the cyclic subgroup H of G generated by p. The elements of H are $\{1, p, p^2, \ldots, p^{k-1}\}$. The cosets of H partition G. Let a_1H, a_2H, $\ldots$, $a_{(q-1)/k}H$ be the $(q-1)/k$ cosets of H that partition G. Then, $X^q - 1 \in \mathbb{F}_p[X]$ can be factorized as follows:

$$\begin{aligned}
X^q - 1 &= (X-1)(X-\omega)(X-\omega^2)\cdots(X-\omega^{q-1}) \\
&= (X-1)\prod_{i=1}^{(q-1)/k}\prod_{j\in a_iH}(X-\omega^j) \\
&= (X-1)\prod_{i=1}^{(q-1)/k}(X-\omega^{a_i})(X-\omega^{a_ip})(X-\omega^{a_ip^2})\cdots(X-\omega^{a_ip^{k-1}}) \\
&\coloneqq (X-1)\prod_{i=1}^{(q-1)/k} g_i(X)
\end{aligned}$$

$g_i(X)$ has degree k and its roots are conjugates of $\omega^{a_i} \in \mathbb{F}_{p^k}$ with respect to $\mathbb{F}_p$ (which are all distinct). Thus, from Lemma 3.6, $g_i(X) \in \mathbb{F}_p[X]$ is the minimal polynomial of ω^{a_i} over $\mathbb{F}_p$, and hence is irreducible over $\mathbb{F}_p$.

□

Thus, the splitting field of $X^q - 1$ over $\mathbb{F}_p$ is $\mathbb{F}_{p^k}$. It is easy to see that $\mathbb{F}_{p^k}$ contains a q-th non residue, since the map $\Phi : \alpha \mapsto \alpha^q$ is not surjective in $\mathbb{F}_{p^k}$ (since for every i, $\Phi(\omega^i) = 1$, where ω is some primitive q-th root of unity).

In order to get our hands on a canonical representation of $\mathbb{F}_{p^k}$, we can factorize $(X^q-1)/(X-1) = X^{q-1} + X^{q-2} + \cdots + X + 1$ over $\mathbb{F}_p$ and pick the lexicographically smallest degree k irreducible factor $h(X)$. Then, $\mathbb{F}_p[X]/h(X)$ is isomorphic to $\mathbb{F}_{p^k}$. Let ω be an element in $\mathbb{F}_{p^k}$ isomorphic to $X \in \mathbb{F}_p[X]/h(X)$. Next, to find a canonical q-th non residue $\alpha \in \mathbb{F}_{p^k}$, we set $\alpha = \omega$ and repeatedly perform $\alpha \leftarrow \sqrt[q]{\alpha}$ (choosing the lexicographically smallest q-th root) until α is a q-th non residue. Algorithm 3 implements the above idea and constructs an irreducible polynomial of degree d. In Line 3 and Line 8, factorization is done using Theorem 3.10. We analyze the correctness and running time of Algorithm 3 in Theorem 5.3.

Theorem 5.3 (Correctness and Runtime of Algorithm 3)**.** *Algorithm 3 pseudo-deterministically constructs an irreducible polynomial of degree d over $\mathbb{F}_p$ and runs in expected time $\tilde{O}(d^4 \log^3 p)$.*

Algorithm 3 Pseudo-deterministic construction of irreducible polynomials over $\mathbb{F}_p$

Input: Degree d

Output: Irreducible polynomial of degree d over $\mathbb{F}_p$

```
1: Initialize arrays H ← [ ], Λ ← [ ]
2: for prime q | d, q ≠ p do
3:     Factorize X^{q-1} + X^{q-2} + ··· + X + 1 = g_1(X) g_2(X) ··· g_{(q-1)/k}(X) over F_p
4:     h(X) ← lexicographically smallest degree k factor among g_1(X), g_2(X), ..., g_{(q-1)/k}(X)
5:     Field arithmetic over F_{p^k} will henceforth be performed over F_p[X]/h(X).
6:     α ← element in F_{p^k} isomorphic to X in F_p[X]/h(X)
7:     while α is a q-th residue do
8:         Factorize X^q − α = (X − β_1)(X − β_2) ··· (X − β_q) over F_{p^k}
9:         α ← lexicographically smallest element among β_1, β_2, ..., β_q in F_{p^k}
10:    end while
11:    Append h(X) to array H and α to array Λ
12: end for
13: Using arrays H and Λ and Theorem 5.1, deterministically construct an irreducible polynomial
    of degree d over F_p.
```

Proof. We need to show that the for loop in Algorithm 3 correctly computes the splitting field of $X^q - 1$ and finds a q-th non residue in the splitting field. Then, Line 13 will correctly output an irreducible polynomial of degree d over $\mathbb{F}_p$ (from Theorem 5.1).

Let k be the smallest integer greater than 0 such that $q \mid p^k - 1$ (k is the order of $p \pmod q$). From Lemma 5.2, $X^{q-1} + X^{q-2} + \cdots + X + 1$ factorizes as $g_1(X)g_2(X)\cdots g_{\frac{q-1}{k}}(X)$ where $g_i(X)$ are degree k irreducible polynomials. Thus, by choosing the lexicographically smallest degree k irreducible factor $h(X)$ of $X^q - 1$, we ensure that the choice of $h(X)$ is canonical. $\mathbb{F}_{p^k} \cong \mathbb{F}_p[X]/h(X)$ is the splitting field of $X^q - 1$ which contains a q-th non residue.

Let $\omega \in \mathbb{F}_{p^k}$ be some primitive q-th root of unity. Suppose α is a q-th residue, and let $\beta \in \mathbb{F}_{p^k}$ such that $\alpha = \beta^q$ (β is a q-th root of α). Then, $\{\beta, \beta\omega, \beta\omega^2, \ldots, \beta\omega^{q-1}\}$ are all q-th roots of α. Thus, as required in Line 8, $X^q - \alpha$ will factorize into linear factors. By ensuring that we pick the lexicographically smallest q-th root of α, we "canonize" the computation of q-th non residue. This "canonization" process is akin to the one Gat and Goldwasser [6, Section 5] used to compute q-th non residue in $\mathbb{F}_p$.

But we still need to ensure that the while loop eventually terminates. Let $p^k - 1 = q^\ell r$, where r is not divisible by q. Note that $\ell \le k \log p$. In each iteration of the while loop, the order of α in $\mathbb{F}_{p^k}^*$ increases by a factor of q. Since the order of α divides $\left|\mathbb{F}_{p^k}^*\right| = q^\ell r$ (by Lagrange's theorem), the while loop will terminate in at most ℓ steps. Thus, for each prime $q \mid d, q \neq p$, the for loop at Line 2 pseudo-deterministically constructs the splitting field $\mathbb{F}_{p^k}$ of $X^q - 1$ and a q-th non residue in $\mathbb{F}_{p^k}$.

Now we analyze the runtime. From Theorem 3.10, equal degree factorization in Line 3 takes $\tilde{O}(q \log^2 p)$. From Lemma 3.9, lexicographically smallest $h(X)$ in Line 4 can be chosen in $O(q \log p)$. The while loop at Line 7 runs at most ℓ times. The factoring step at Line 8 takes $\tilde{O}(qk^2 \log^2 p)$ (using Theorem 3.10) and the lexicographically smallest q-th root can be picked in $O(qk \log p)$ time. Thus, the while loop takes $\tilde{O}(\ell q k^2 \log^2 p)$. Since $\ell \le k \log p$ and $k < q$, the running time of the while loop can be upper bounded by $\tilde{O}(q^4 \log^3 p)$. Thus the overall running time of each iteration of the for loop is $\tilde{O}(q^4 \log^3 p)$. So we can upper bound the running time of the entire for loop by $\tilde{O}(d^4 \log^3 p)$. Since the running time of Line 13 is also upper bounded by $\tilde{O}(d^4 \log^3 p)$ (from Theorem 5.1), the overall running time of the algorithm is $\tilde{O}(d^4 \log^3 p)$. □

We end this section by completing the proof of Theorem 1.3.

Proof of Theorem 1.3. Algorithm 2 pseudo-deterministically constructs irreducible polynomial of degree d over $\mathbb{F}_{p^k}$. From Theorem 4.1, it takes time $T(dk, p) + \tilde{O}(dk^3 \log p)$, where $T(dk, p)$ is time taken for the sub-routine which constructs degree dk irreducible polynomial over $\mathbb{F}_p$. We use Algorithm 3 to implement this sub-routine, which from Theorem 5.3 takes time $\tilde{O}(d^4 k^4 \log^3 p)$. Thus, the overall running time is $\tilde{O}(d^4 k^4 \log^3 p)$. Let $q = p^k$. Thus, we have given a pseudo-deterministic algorithm for constructing irreducible polynomials of degree d over $\mathbb{F}_q$ in expected time $\tilde{O}(d^4 \log^4 q)$. □

6 Conclusion

We have shown an efficient pseudo-deterministic algorithm for constructing irreducible polynomials of degree d over finite field $\mathbb{F}_q$. It is natural to ask if this algorithm can be derandomized to get a fully deterministic algorithm. Since our approach heavily relies on fast randomized polynomial factoring algorithms, and no efficient deterministic factoring algorithms are known, it is unclear how to derandomize it using the above approach. In fact, we don't even know how to deterministically construct a quadratic non residue modulo p (p is prime).

Another interesting question is to compare the hardness of deterministically factoring polynomials and deterministically constructing irreducible polynomials over finite fields. As mentioned earlier, Shoup [12] had showed that constructing irreducible polynomials over finite fields can be efficiently (and deterministically) reduced to factoring polynomials. This suggests that factoring polynomials is as hard as constructing irreducible polynomials. But what about the other direction? Would we be able to factor polynomials efficiently if we could construct irreducible polynomials?

The answer is affirmative in the quadratic case. Suppose we are given a quadratic non residue β modulo p. Then we can compute the square roots of any quadratic residue α module $\mathbb{F}_p$. In other words, given an irreducible polynomial $X^2 - \beta$, we can factorize $X^2 - \alpha$. This can be achieved using the Tonelli-Shanks [11, 14] algorithm for computing square roots modulo p. However, this technique does not easily generalize to higher degrees d, so there isn't enough evidence to confirm that constructing irreducible polynomials is as hard as factoring polynomials in general. We believe this is an interesting open question that can shine more light on the complexity of both these problems.

Gat and Goldwasser [6] highlighted the open problem of pseudo-deterministically constructing n-bit prime numbers, which still remains unsolved. Chen et al. [4] solved this problem but with the caveat that their algorithm works in the infinitely often regime. Their algorithm is based on complexity theoretic ideas. In this paper, we gave a pseudo-deterministic algorithm for constructing irreducible polynomials, which leverages the structure of irreducible polynomials. Perhaps similarly one could hope to get an efficient pseudo-deterministic algorithm for constructing primes using some number theoretic approaches.

Acknowledgements

The author would like to thank Mrinal Kumar and Ramprasad Saptharishi for introducing him to the question of pseudo-deterministic construction of irreducible polynomials and for the many insightful discussions along the way.

References

[1] Leonard M. Adleman and Hendrik W. Lenstra Jr. Finding irreducible polynomials over finite fields. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, STOC '86, pages 350–355, New York, NY, USA, 1986. ACM. `doi:10.1145/12130.12166`.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. URL: `http://www.jstor.org/stable/3597229`.

[3] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981. URL: `http://www.jstor.org/stable/2007663`.

[4] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1261–1270, Los Alamitos, CA, USA, nov 2023. IEEE. `doi:10.1109/FOCS57990.2023.00074`.

[5] Jean-Marc Couveignes and Reynald Lercier. Fast construction of irreducible polynomials over finite fields. *Israel Journal of Mathematics*, 194(1):77–105, mar 2013. URL: `http://link.springer.com/10.1007/s11856-012-0070-8`, `doi:10.1007/s11856-012-0070-8`.

[6] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, TR11-136(TR11-136), oct 2011. URL: `https://eccc.weizmann.ac.il/report/2011/136`, `arXiv:TR11-136`.

[7] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011. `arXiv:https://doi.org/10.1137/08073408X`, `doi:10.1137/08073408X`.

[8] H. W. Lenstra. Finding isomorphisms between finite fields. *Mathematics of Computation*, 56(193):329–347, 1991. URL: `https://www.ams.org/mcom/1991-56-193/S0025-5718-1991-1052099-2/`, `doi:10.1090/S0025-5718-1991-1052099-2`.

[9] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, Cambridge, 2 edition, 1994. URL: `https://www.cambridge.org/core/books/introduction-to-finite-fields-and-their-applications/84CD611976678E270A2146652E1773D6`, `doi:10.1017/CBO9781139172769`.

[10] Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9(2):273–280, 1980. `arXiv:https://doi.org/10.1137/0209024`, `doi:10.1137/0209024`.

[11] Daniel Shanks. Five number-theoretic algorithms. In *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972)*, volume No. VII of *Congress. Numer.*, pages 51–70. Utilitas Math., Winnipeg, MB, 1973.

[12] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 283–290. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21944.

[13] Terence Tao, Ernest Croot III, and Harald Helfgott. Deterministic methods to find primes. *Math. Comput.*, 81(278):1233–1246, 2012. doi:10.1090/S0025-5718-2011-02542-1.

[14] Alberto Tonelli. Bemerkung über die auflösung quadratischer congruenzen. *Nachrichten von der Königl. Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen*, 1891:344–346, 1891. URL: http://eudml.org/doc/180329.

[15] Joachim von zur Gathen and Victor Shoup. Computing frobenius maps and factoring polynomials. *Comput. Complex.*, 2(3):187–224, sep 1992. doi:10.1007/BF01272074.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il