FINGERPRINTING BY RANDOM POLYNOMIALS

by

Michael O. Rabin

TR-15-81

FINGERPRINTING BY RANDOM POLYNOMIALS

by

Michael O. Rabin
Department of Mathematics
The Hebrew University of Jerusalem

and

Department of Computer Science
Harvard University

Abstract

Randomly chosen irreducible polynomials $p(t) \in Z_2[t]$ are used to "fingerprint" bit-strings. This method is applied to produce a very simple real-time string matching algorithm and a procedure for securing files against unauthorized changes. The method is provably efficient and highly reliable for every input.

This research was supported in part by NSF Grant MCS-80-12716 at the University of California at Berkeley.

Fingerprinting by Random Polynomials

by

Michael O. Rabin

1. Introduction

Prime numbers are used in several contexts to yield efficient randomized algorithms for various problems [1,4]. In these applications a randomly chosen prime p is used to "fingerprint" a long character-string by computing the residue of that string, viewed as a large integer, modulo p.

This method requires performing fixed-point arithmetic on k-bit integers, where $k = \lceil \log_2 p \rceil$, or at least addition/subtraction on such integers [4].

We propose using a randomly chosen irreducible (prime) polynomial $p(t) \varepsilon Z_2[t]$ of an appropriate small degree k instead of the prime integer p. It turns out that it is very easy to effect a random choice of an irreducible polynomial. The implementation of $\bmod p(t)$ arithmetic requires just length-k shift registers and the exclusive-or operation on k-bit vectors. These operations are fast, involve simple circuits, and in VLSI require little chip area.

The applications given here are: a real-time string matching algorithm; detection of unauthorized changes in a file. The method obviously extends into other areas.

Polynomial modular computations are used in algebraic error correction codes. In these applications one carefully constructs a specific polynomial which is designed to facilitate coding and decoding. Under assumption of a random distribution of all possible error patterns, the code will detect/correct most occurrences of up to ℓ errors for some fixed ℓ. In the style of [6], we turn the tables around and instead of applying a fixed algorithm to (hopefully) randomly distributed inputs, we construct a randomizing algorithm which is efficient on every input. By randomizing the choice of the irreducible polynomial $p(t)$, we obtain a provably highly dependable and efficient algorithm for <u>every</u> instance of the string matching problem to be solved; we successfully protect every file against any deliberate modification, etc. The mathematically provable efficacy of our method is of particular significance for this last application.

2. Counting and Choosing Irreducible Polynomials

Even though any degree k can be used, implementation is most convenient when k is prime. Thus in practice we can use $k = 17, 19, \ldots, 31, 61$, etc.

<u>Lemma 1</u>. Let k be prime. The number of irreducible polynomials $p(x) = x^k + a_{k-1}x^{k-1} + \ldots + a_0 \in Z_2[x]$, is $(2^k - 2)/k$.

<u>Proof</u>. Let $GF(2^k) = E$ be the Galois field with 2^k elements. Every irreducible polynomial $p(x) \in Z_2[x]$ of degree k has exactly k roots in E, and since $1 < k$ these roots are in $E - Z_2$.

Let $\alpha \in E - Z_2$, and let $1 < m$ be the degree of the irreducuble polynomial $q(x) \in Z_2[x]$ that α satisfies. Then $[Z_2(\alpha): Z_2] = m$ and hence $m|k$ (see [3]). Since k is prime, $m = k$. I.e., every $\alpha \in E - Z_2$ is a root of an irreducible polynomial of degree k.

Thus the set $E - Z_2$ is partitioned into sets of k elements,

$f \in Z_2[x]$ of degree k, and all such polynomials are obtained in this way. It follows that the number of these polynomials is $(2^k-2)/k$. ■

Consider a fixed prime k, say $k = 31$, how do we randomly choose an irreducible polynomial $t^{31} + b_1 t^{30} + \ldots \in Z_2[t]$? Wé shall do this by calculating within the field $E = GF(2^k)$ of 2^k elements.

To this end we need <u>one</u> irreducible polynomial $f(x) = x^k + a_1 x^{k-1} + + \ldots \in Z_2[x]$. The elements of E will be the k-tuples $\gamma = (c_1, \ldots, c_k)$, $c_i \in Z_2$. The addition of two elements is component-wise. To find the product $\gamma \cdot \delta$, where $\delta = (d_1, \ldots, d_k)$, calculate (in $Z_2[x]$) the residue

$$e_1 x^{k-1} + \ldots + e_k \equiv (c_1 x^{k-1} + \ldots + c_k)(d_1 x^{k-1} + \ldots + d_k) \bmod f(x);$$

then $\gamma \cdot \delta = (e_1, \ldots, e_k)$.

Details of how to computationally implement the arithmetic of a finite field can be found in [7], where an efficient method for finding irreducible polynomials of any degree n is also given. We assume that irreducible polynomials $f_2(x)$, $f_3(x)$, ... $f_{31}(x), \ldots,$ of small prime degrees are tabulated once and for all.

We shall now effect a random choice of an irreducible polynomial $p(t) = t^{31} + \ldots \in Z_2[t]$. We use the indeterminate t to distinguish these polynomials from the fixed polynomials $f_2(x)$, $f_3(x), \ldots$. Choose randomly an element $\gamma = (c_1, \ldots, c_{31}) \in GF(2^{31}) - Z_2$. Namely, randomly generate a sequence of 31 bits and if it happens to be $(0,0,\ldots,0)$ or $(0,0,\ldots,0,1)$ discard it.

The element γ satisfies, by the proof of Lemma 1, a unique irreducible polynomial $p(t) = t^{31} + b_1 t^{30} + \ldots + b_{31} \in Z_2[t]$. To find it, compute in $GF(2^{31})$ (using $f_{31}(x)$) the powers

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.