Public Key Encryption and Signature Schemes Based on Polynomials over $\mathbb{Z}_n$

Jörg Schwenk[1] and Jörg Eisfeld[2]

[1] Deutsche Telekom AG, Forschungszentrum, Am Kavalleriesand 3, D-64295 Darmstadt, Germany, schwenk@fz.telekom.de
[2] Justus-Liebig-Universität Gießen, Mathematisches Institut, Arndtstr. 2, D-35392 Gießen, Germany, joerg.eisfeld@math.uni-giessen.de

Abstract. The problem of computing roots of a polynomial over the ring $\mathbb{Z}_n$ is equivalent to factoring n. Starting from this intractable problem we construct a public key encryption scheme where the message blocks are encrypted as roots of a polynomial over $\mathbb{Z}_n$ and a signature scheme where the signature belonging to a message is a (set of) root(s) of a polynomial having the message blocks as coefficients. These schemes can be considered as extensions of Rabin's encryption and signature scheme. However, our signature scheme has some new properties: a short signature can be generated for a long message without using a hash function, and the security features of the scheme can be chosen either to be similar to those of the RSA scheme or to be equivalent to those of Rabin's scheme.

1 Introduction

In this paper we investigate an intractable problem related to the problem of factoring. This problem can be stated as follows:

Root Finding Problem (RFP): *Compute one (all) root(s) of a polynomial $f(x)$ over the ring $\mathbb{Z}_n$ where $n = pq$ is the product of two large primes.*

For randomly chosen polynomials the RFP is equivalent to the problem of factoring n whenever $f(x)$ has at least two different roots. If $f(x)$ has exactly one root, the root finding problem seems to be related to the problem of decrypting an RSA ciphertext. Section 2 summarizes some mathematical results on the RFP.

In Sect. 3 we present a public key encryption scheme the security of which is based on the difficulty of the root finding problem: A message is divided into blocks and a certain redundancy is added to these blocks. The result of this operation is interpreted as an integer less or equal to n. These numbers $z_1, \ldots, z_k$ are then encrypted as roots of the polynomial $f(x) := (x - z_1) \cdots (x - z_k) \bmod n$. The public key used for this operation is the modulus n. Decryption can be done by using the private key (p, q) to compute roots of the polynomials $f(x) \bmod p$ and $f(x) \bmod q$ in the corresponding finite fields, e.g. with the algorithm described by Ben-Or in [BO81], and by combining these results via the Chinese remainder theorem. The chosen redundancy scheme will then help to find the k correct roots out of the k^2 solutions.

In Sect. 4 a signature scheme is described where the roles of coefficients and roots are interchanged: The blocks of the message are interpreted as coefficients of a polynomial. The signature z of the message is a root of this polynomial. However, since not all the polynomials in $\mathbb{Z}_n[x]$ have a root, some randomization must be used in order to generate a signature, e.g. by choosing the coefficient of x^0 randomly. The feasability of the signature scheme is guaranteed by Thm. 3. This theorem and the corollaries following it give the number of polynomials in $\mathbb{Z}_n[x]$ which have at least one root (at least two roots, resp.).

The encryption and signature scheme presented in this paper are extensions of ideas of Rabin [Rab80], but due to the more general setting they are more flexible. By choosing a randomized polynomial with the appropriate number of roots, the security features of our signature scheme can be chosen either to be equivalent to those of Rabin's scheme or to be similar to those of the RSA scheme.

Furthermore, our signature scheme has the property that we can generate short signatures for long messages without using a hash function. As far as the authors know no other signature scheme with this property exists.

2 Some Properties of Polynomials over $\mathbb{Z}_n$

The root finding problem is a new intractable problem based on the problem of factoring integers. Other well-known problems based on factoring are the problems of computing square roots of quadratic residues and breaking the RSA public key cryptosystem. In this paper, n always denotes an integer that is the product of two large primes p and q: $n = pq$.

Finding a root of a randomly chosen polynomial $f \in \mathbb{Z}_n[x]$ is equivalent to factoring n if $f(x)$ has at least two different roots. This will be proved in Thm. 2. If $f(x)$ has only one root over $\mathbb{Z}_n$, then the root-finding problem includes the problem of decrypting an RSA ciphertext as a special case: To decrypt $c = m^e$, we have to find the unique root of the polynomial $x^e - c \in \mathbb{Z}_n[x]$.

If we know the factorization (p, q) of n, the problem of finding a root of a polynomial $f \in \mathbb{Z}_n[x]$ can be reduced to the (easy) problem of computing roots in the finite fields GF(p) and GF(q) [BO81]. The results can be combined to construct solutions of the root-finding problem by using the extended Euclidean algorithm and the Chinese remainder theorem. Details of this construction can be found in the appendix.

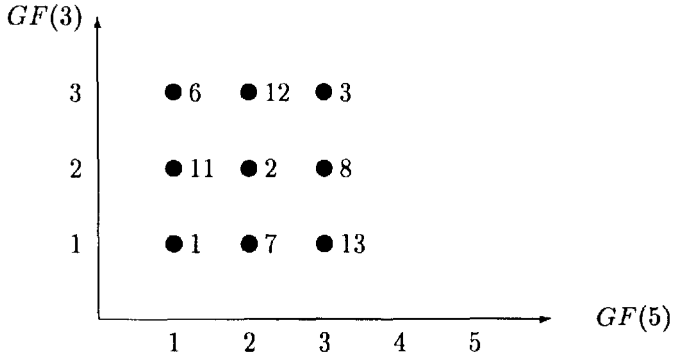
In order to get a better understanding of the structure of the set of roots of a polynomial defined over $\mathbb{Z}_n$, and in order to be able to prove Thm. 2 given below, we need the following definitions.

Definition 1. Let $f(x)$ be a polynomial over $\mathbb{Z}_n$.
(1) Two roots z_1 and z_2 of $f(x)$ are said to be *in the same row* (resp. *in the same column*) if they are equivalent modulo q (resp. modulo p).

(2) A set $\{z_1, \ldots, z_t\}$ of roots of $f \in \mathbb{Z}_n[x]$ is called a *partial transversal* if no two of its elements are in the same row or column. A *transversal* is a maximal partial transversal.

The following examples illustrate the concepts of transversal and being in a row or column.

Example 1. (1) The set $\{u_1, \ldots, u_t\}$ is a transversal of the set of roots of $g(x) := (x - u_1)(x - u_2) \cdots (x - u_t) \bmod n$ if the u_i are such that $(u_i)_p \neq (u_j)_p$ and $(u_i)_q \neq (u_j)_q$ for $i \neq j$.
(2) In the picture given below the set of roots of the polynomial $g(x) = (x-1)(x-2)(x-3) \pmod{15}$ is given. It has six transversals $\{1,2,3\}$, $\{6,7,8\}$, $\{11,12,13\}$, $\{1,12,8\}$, $\{6,2,13\}$ and $\{11,7,3\}$.



The proofs of the following theorems can be found in the appendix.

Theorem 2. *The problem of computing roots of polynomials over $\mathbb{Z}_n$ which have at least two different roots is equivalent to the the problem of factoring n.*

If we choose a polynomial $f(x)$ at random, e.g. by choosing its coefficients at random, we cannot be sure if this polynomial has a root or not. E.g. if $f(x)$ is irreducible either over GF(p) or over GF(q), then $f(x)$ has no roots modulo n. The following theorem gives the percentage of polynomials of degree k which have exactly m roots over the finite field GF(p).

Theorem 3. *The fraction of polynomials of degree d over* GF(q) *which have exactly m roots converges for $q \to \infty$ to $\frac{1}{m!} \cdot \sum_{i=0}^{d-m} (-1)^i \cdot \frac{1}{i!}$. By increasing $d - m$ this number converges very quickly to $\frac{1}{m!} \cdot e^{-1}$.*

The following corollaries guarantee that the generation of signatures in the signature scheme described in Sect. 4 is feasible. They give the probability that a randomly chosen polynomial will have al least one root (at least two roots, resp.).

Corollary 4. *(1) The fraction of polynomials of degree d over* $\mathrm{GF}(q)$ *which have no roots converges for $q \to \infty$ to $\sum_{i=0}^{d}(-1)^i \cdot \frac{1}{i!}$. By increasing d this number converges very quickly to $e^{-1} \approx 0.36788$.*
(2) The fraction of polynomials of degree d over $\mathrm{GF}(q)$ *that have at most one root converges for $q \to \infty$ to $2\left(\sum_{i=0}^{d}(-1)^i \cdot \frac{1}{i!}\right) - (-1)^d \cdot \frac{1}{d!}$. By increasing d, this number converges very quickly to $2e^{-1} \approx 0.73576$.*

Corollary 5. *(1) For large d and q the fraction of polynomials of degree d over* $\mathrm{GF}(q)$ *that have at least one root (at least two roots, resp.) can be approximated by $1 - e^{-1}$ ($1 - 2e^{-1}$, resp.).*
(2) For large d and n the fraction of polynomials of degree d over $\mathbb{Z}_n$ that have at least one root can be approximated by $(1 - e^{-1})^2$.
(3) For large d and n the fraction of polynomials of degree d over $\mathbb{Z}_n$ that have at least two roots can be approximated by $1 - 2e^{-1}$.

3 Encrypting a Message as Roots of a Polynomial

In the preceding section we have established an intractable problem: computing roots of a polynomial defined over $\mathbb{Z}_n$. It is quite natural to use this problem to encrypt data, in the same way as Rabin used the problem of computing square roots for his encryption system [Rab80].

The public key of this encryption scheme is the modulus n, the private key is the factorization (p, q) of n.

Encryption function E(m)

1. Split the message m into blocks $m = \overline{m_1} || \ldots || \overline{m_k}$.
2. Add some redundany to these blocks to make them recognizable and to order them. Let $m_1, \ldots, m_k$ denote the result of this operation.
3. Interpret $m_1, \ldots, m_k$ as numbers less or equal to n.
4. Compute $f(x) := (x - m_1)\cdots(x - m_k) \bmod n = x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0$.
5. Transmit the ciphertext $c = E(m) := (a_{k-1}, \ldots, a_1, a_0)$.

The necessity of the second step, making the numbers *recognizable*, will become clear when we look at the decryption function.

Decryption function D(c)

1. Reconstruct the polynomial $f(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0$ from the ciphertext $c := (a_{k-1}, \ldots, a_1, a_0)$.
2. Compute the roots $v_1, \ldots, v_k$ of $f_p(x) := f(x) \bmod p$ over $\mathrm{GF}(p)$ and $w_1, \ldots, w_k$ of $f_q(x) := f(x) \bmod q$ over $\mathrm{GF}(q)$.
3. The extended Euclidean algorithm yields $1 = \lambda p + \mu q$. Use this equation in the Chinese remainder theorem to compute the k^2 roots $z_{ij} = w_i \lambda p + v_j \mu q \bmod n$ of $f(x)$ in $\mathbb{Z}_n$.
4. With high probability, exactly k of these roots will fit into the given redundancy scheme. Arranging these roots in the right order and removing the redundancy will give the original message $m := D(c)$.

Security of the encryption scheme

Public Key property. Computing a root of a polynomial over $\mathbb{Z}_n$ is equivalent to factoring n by Thm. 2. But we cannot apply Thm. 2 directly since we are not dealing with random polynomials, but only with a certain subset of $R[x] \subseteq \mathbb{Z}_n[x]$. It consists of those polynomials which have at least one transversal satisfying the redundancy scheme.

However, if the redundancy scheme is weak in the sense that this subset is large, we could use an oracle that outputs a root which fits into this redundancy scheme exactly like the more general oracle in the proof of Thm. 2. This would just increase the number of trials by a factor of $\frac{n^d}{|R_d[x]|}$, where $R_d[x]$ consists of the polynomials of degree d from $R[x]$.

There is a clear relation between the strength of the redundancy scheme and our ability to prove equivalence to the factoring problem. But if we choose the redundancy scheme too weak, the second plaintext attack described below can break the system completely.

It should be clear that is not possible to break the system (i.e. to factor the public key n) by simply choosing some message blocks $m_1, \ldots, m_k$ and the trying to use these roots and the corresponding polynomial $f(x)$ to factor n, because with overwhelming probability the numbers $m_1, \ldots, m_k$ will form a transversal (cf. proof of Thm. 2).

Second plaintext attack. If an attacker chooses the message blocks $m_1, \ldots, m_k$ and then manages to make the owner of the private key decrypt the ciphertext c for him, the redundancy scheme must be strong enough to prevent this attack. I.e. regardless how the attacker chooses $m_1, \ldots, m_k$, the probability that there is a root different from $m_1, \ldots, m_k$ satisfying the redundancy scheme is negligible.

GCD attack. If two messages m, m' are encrypted with the same public key n, and if these messages have one message block m_i in common, then it may be possible to recover m_i by calculating the greatest common divisor of the corresponding polynomials f and f'. However, this does not mean that the system is broken. Observe that information which is common in many messages can not be considered to be very confidential (e.g. the name of the owner of the public key).

4 Signing a Message by Computing a Root

A less obvious application of polynomials over $\mathbb{Z}_n$ is the signature scheme described in this section. We get a signature scheme based on the RFP if we interchange the roles of coefficients and roots in the encryption scheme of the last section. In this scheme, blocks of a message are interpreted as coefficients of a polynomial. The signature of this message consists of one or more roots of this polynomial; in case of two or more roots, these roots must of course be part of a transversal.

Signature function Sig(m)

1. Divide the message m into blocks $a_{k-1}, \ldots, a_1$.
2. Interpret these blocks as numbers less or equal to n.
3. Choose a random number $\overline{a_0} \leq n$. Add some redundancy to this number and use the result a_0 as the coefficient of x^0.
4. Compute one or more roots $r_1, \ldots, r_t$ $(1 \leq t \leq k)$ which are part of one transversal of the polynomial $f(x) := x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 \bmod n$. If $f(x)$ has no roots or doesn't have enough roots, then repeat step (3).
5. Add the signature $s = Sig(m) = (a_0, r_1, \ldots, r_t)$ to the message m.

If we add at least two roots from a transversal, then the problem of breaking this signature scheme is equivalent to the problem of factoring.

Verification function Ver(m,s)

1. Reconstruct the polynomial $f(x) := x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0$ from m and $s = (a_0, r_1, \ldots, r_t)$.
2. For $i = 1, \ldots, t$ compute $f(r_i) \bmod n$ and check whether all these values equal 0. If this is the case, the signature is accepted, otherwise rejected.

The signature can be made much shorter than the message to be signed: it is sufficient to use one root r, and the random part of the coefficient a_0 can be made as short as it is possible without allowing for an exhaustive search attack. E.g., this means if factoring of 768 Bit numbers is considered to be intractable, a short signature could consist of $768 + 64$ Bit, for messages of arbitrary length. *To produce such a short signature, no hash function is necessary.*

Choosing a_0 at random is not the only way to make the signature scheme work. It is only necessary to use some randomization for generating messages, but there are various other possibilities for doing this, e.g. choosing some bits of each coefficient at random. However, this randomization has to be integrated in the signature function in such a way that the attacks described below will not work.

Security of the signature scheme

Any direct attack on the signature scheme, i.e. trying to directly compute a signature for a given message m, is in difficulty either equivalent to factoring n or related to the security of the RSA signature scheme If the polynomial $f(x)$ has at least two different roots, Thm. 2 applies and guarantees the equivalence with the factoring problem. If $f(x)$ has only one root, our scheme includes the computation of a RSA signature as a special case: the RSA signature of m is the unique root of $x^e - m \pmod{n}$.

We can now state a number of possible attacks on our scheme and how they can be made impossible. In describing the security features of our signature scheme we make use of the classification of attacks and features given in [GMR88].

Existential forgery by multiplication with a linear polynomial. It is possible to get a valid pair (m, s) by first choosing a message $\widehat{m}$, computing the corresponding polynomial $\widehat{f}(x)$ of degree $k-1$ and multiplying this polynomial by $x-r$: $f(x) := \widehat{f}(x)(x-r) \bmod n$. Then the coefficients $(a_{k-1}, \ldots, a_1)$ of $f(x)$ form the message m, and (a_0, r) is a valid signature for this message. However, if we proceed in this way, it is difficult to determine $\widehat{m}$ and r in such a way that the resulting message m makes any sense: If we could fix the message (m, a_0) in advance and then determine $\widehat{m}$ and r, this would give an algorithm for solving the root finding problem.

It should be noted that a similar attack works for the RSA signature scheme: Choose a number r and compute $r^e \bmod n$. Then $(m, s) := (r^e, r)$ is a valid message-signature pair.

Universal forgery by choosing a pair (a_0, s). If we want to sign a message $(a_{k-1}, \ldots, a_1)$, we may simply use the polynomial $g(x) := x^k + a_{k-1}x^{k-1} + \cdots + a_1 x$ to compute a_0 from s. Choose r randomly and let $s := r$ and $a_0 := -g(r)$. Then the polynomial $f(x) = g(x) + a_0 = g(x) - g(r)$ has $s = r$ as a root.

To make this attack impossible it is necessary to guarantee that the root r is calculated from the whole set of coefficients including a_0. This can be achieved by e.g. requiring that the coefficient a_0 fits into a given redundancy scheme, or by computing a_0 from the other coefficients.

Total breakability by a directed Chosen Message Attack. An attacker may choose the message $(a_{k-1}, \ldots, a_1)$ and a coefficient $\widehat{a_0}$ in such a way that he knows one root of the corresponding polynomial. He may then try to make the owner of the private key sign the message with the given $\widehat{a_0}$ in order to get a different root in the same row or column.

To avoid this attack, it must be guaranteed that the coefficient a_0 (or any other probabilistic element of the scheme) is choosen by the signer of the message only. In this case the probability that $a_0 = \widehat{a_0}$ is negligible.

Another method to make such an attack impossible is to require that the coefficient a_0 is chosen in such a way that the polynomial $f(x)$ has exactly one root over $\mathbb{Z}_n$. The fraction of polynomials with this property is large enough to guarantee the feasability of this approach (cf. Cor. 5).

Redundancy Schemes

During our previous discussions, we used the term "redundancy scheme" in a very broad sense. A redundancy scheme in this sense could consist of fixing some of the bits of the a_i, or of applying an error correcting code or a cryptographic hash function on the $\overline{a_i}$. There may be some applications where some redundancy scheme is already given, e.g. an error correcting code is needed for the reliable transmission of the coefficients. The only requirement on the redundancy scheme used is that the number of bits which will be determined through this scheme is large enough to prevent an exhaustive search attack; this number may e.g. range from 64 to 128 bits.

5 Performance

The most time consuming task in the two schemes presented is to compute roots of polynomials defined over GF(p)[x]. A fast probabilistic polynomial time algorithm for computing roots of polynomials of degree d in GF(p) was proposed by Rabin [Rab80a]. The investigation of its complexity was later refined by Ben-Or [BO81]: he proved that Rabin's algorithm needs only $O\left(d(\log d)^2 \log^2 d \log p\right)$arithmetical operations in GF(p) to compute all the roots of a polynomial of degree d.

When using Rabin's algorithm to compute roots of a polynomial in a finite field, in the worst case the same number of arithmetical operations is needed for computing only one root or all the roots. So the complexity for decryption of a ciphertext and for the computation of a signature only differ by a constant factor, namely the expected number of times one has to choose a random $\overline{a_0}$ to be able to compute a signature.

The following table gives a comparison of the performance of our schemes with the RSA variant where the public exponent is always chosen to be equal to 3. The advantages of our scheme are more obvious if compared to the normal RSA scheme.

	Polynomial Schemes	RSA (d blocks) with $e = 3$
Encryption	$O(d)$	$O(d)$
Decryption	$O\left(d(\log d)^2 \log^2 d \log p\right)$	$O(d \log n)$
Generationof a signature	$O\left(d(\log d)^2 \log^2 d \log p\right)$	$O(d \log n)$
Verification of a signature	$O(d)$	$O(d)$

6 Open Problems

An interesting open problem is to investigate in more detail the difficulty of computing the unique root of a polynomial that has only one root. This could throw some light on the famous open question about the security of the RSA scheme in relation to the difficulty of factoring integers.

The following variant of our schemes has been proposed by one of the referees: Take $f(x) := a_k x^k + \ldots + a_1 x + a_0 \bmod n$ as Alice's public key, and let (p, q) be her private key. Then a message m can be encrypted as the polynomial $c(x) := f(x) - f(m)$, where m is a root of $c(x)$. The message m could be signed by computing a root of either $c(x)$ or of $\bar{c}(x) := f(x) - m$. The properties of the polynomial $c(x)$ have to be investigated. This approch has as disadvantage the length of the ciphertext to be transmitted, but may be useful as a signature scheme.

7 Acknowledgements

The authors thank A. Beutelspacher for valuable comments on the schemes proposed above, and A. Scheerhorn for performing some important statistical tests with the computer algebra system AXIOM which prepared the grounds for Thm. 3. We also thank the referees for their comments on a previous version of the paper, which helped to improve the quality of this paper.

References

[BO81] M. Ben-Or, *Probabilistic Algorithms in Finite Fields.* Proc. IEEE FOCS 1981, pp. 394-398.

[FN93] A. Fiat and M. Naor, *Broadcast Encryption.* Pre-Proceedings of CRYPTO '93, 39.1-39.10.

[GMR88] S. Goldwasser, S. Micali and R. L. Rivest, A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. Vol. 17 No. 2 (1988), pp. 281-308.

[Rab80] M. O. Rabin, *Digital Signatures and Public-Key Functions as Intractable as Factorization.* MIT/LCS/TR-212, Jan 1979. Cited after B. Schneier.

[Rab80a] M. O. Rabin, *Probabilistic Algorithms in Finite Fields.* SIAM J. Comput. 9(1980), pp. 273-280.

[Sha93] A. Shamir, *On the Generation of Multivariate Polynomials which are Hard to Factor.* Preprint.

[Sch93] B. Schneier, *Applied Cryptography.* Wiley 1993.

8 Appendix: Proofs of the Theorems in Section 2

For any $a, p \in \mathbb{Z}$ let a_p denote $a \bmod p$. If p and q are different primes, the extended Euclidean algorithm computes integers λ and μ such that

$$1 = \lambda p + \mu q.$$

This equation is used in the Chinese remainder theorem to construct a solution,

$$a = a_q \lambda p + a_p \mu q,$$

which is unique in $\mathbb{Z}_n$, of the system of congruences

$$a \equiv a_p \pmod p \qquad \text{and} \qquad a \equiv a_q \pmod q.$$

Lemma 6. *The integer a is a root of the polynomial $f \in \mathbb{Z}_n[x]$ $(n = pq)$ if and only if $f(a_p) \equiv 0 \pmod p$ and $f(a_q) \equiv 0 \pmod q$.*

Proof. Reducing modulo p (modulo q) is a ring homomorphism from $\mathbb{Z}_{pq}$ to $\mathbb{Z}_p$ $(\mathbb{Z}_q)$, so it is clear that from $f(a) \equiv 0 \pmod n$ the two reduced equations follow.
On the other hand, $f(a) \equiv f(a_p) \equiv 0 \pmod p$ and $f(a) \equiv f(a_q) \equiv 0 \pmod q$, which gives $f(a) \equiv 0 \pmod n$. □

Lemma 7. *(1) The map* $\mathrm{GF}(p) \times \mathrm{GF}(q) \to \mathbb{Z}_{pq}$ *given by* $(a_p, a_q) \mapsto a \equiv a_q\lambda p + a_p\mu q \pmod n$ *is a ring isomorphism.*
(2) If

$$a \equiv a_q\lambda p + a_p\mu q \pmod n \quad \textit{and} \quad b \equiv b_q\lambda p + b_p\mu q \pmod n$$

are roots of $f \in \mathbb{Z}_n[x]$, *then*

$$c \equiv a_q\lambda p + b_p\mu q \pmod n \quad \textit{and} \quad d \equiv b_q\lambda p + a_p\mu q \pmod n$$

are also roots of f.
(3) If k_p *and* k_q *are the numbers of roots of* $f(x)$ *in* $\mathrm{GF}(p)$ *and* $\mathrm{GF}(q)$ *resp., then* k_pk_q *is the number of roots of* $f \in \mathbb{Z}_n[x]$. *In particular, the number of roots of* $f(x) \pmod n$ *is at most* $(\deg f)^2$.

Proof. (1) The solution given by the Chinese remainder theorem is unique in $\mathbb{Z}_n$.
(2) We have $f(c) \equiv f(a_q) \equiv f(a) \equiv 0 \pmod q$ and $f(c) \equiv f(b_p) \equiv f(b) \equiv 0 \pmod p$. Combining this we get $f(c) \equiv 0 \pmod n$, and the same argument applies to d.
(3) This follows from (1), Lemma 1 and the fact that in the fields $\mathrm{GF}(p)$ and $\mathrm{GF}(q)$ the polynomial f has at most $\deg f$ roots. □

If we know the factorization of n, we can compute all roots of $f(x)$ mod n: There are probabilistic polynomial time algorithms for computing roots of polynomials of degree d [BO81] that need $O\left(d(\log d)^2 \log^2 d \log p\right)$ arithmetical operations in $\mathrm{GF}(p)$, and from the two lemmas above we get all the roots in $\mathbb{Z}_n$.

Lemma 8. *Let* $g(x) := (x - u_1)\cdots(x - u_t)h(x)$ *a polynomial from* $\mathbb{Z}_n[x]$, *where* $\{u_1, \ldots, u_t\}$ *is a transversal of the set of roots of* $g(x)$.
(1) If we know two roots of $g(x)$ *that are in the same row or in the same column, then we can easily compute the factorization of* n.
(2) If $\{z_1, \ldots, z_t\}$ *is a transversal different from* $\{u_1, \ldots, u_t\}$, *then* $g(x) \equiv (x - z_1)\cdots(x - z_t)h(x) \pmod n$.

Proof. (1) Let z_i and z_j be in the same row, i. e. $(z_i)_q = (z_j)_q$. Then $q = \gcd(n, z_i - z_j)$.
(2) We have $(x - u_1)\cdots(x - u_t) \cdot h(x) \equiv (x - (u_1)_p)\cdots(x - (u_t)_p) \cdot h(x) \equiv (x - (z_1)_p)\cdots(x - (z_t)_p) \cdot h(x) \equiv (x - z_1)\cdots(x - z_t) \cdot h(x) \equiv g(x) \pmod p$. A similar equation holds for q, so the statement is proved. □

Proof of Theorem 2. If we know the factorization of n, we can compute the roots of $f(x)$ in the finite fields $\mathrm{GF}(p)$ and $\mathrm{GF}(q)$ in polynomial time [BO81] and combine the results via Lemma 1 and 2.

Now lets assume that we have an oracle that, when fed with a polynomial $f(x) \in \mathbb{Z}_n[x]$, which has at least two different roots, outputs one root of this polynomial. We can use this oracle to factor n in the following way.

First we choose $a, b \in \mathbb{Z}_n$ and $g(x) \in \mathbb{Z}_n[x]$ at random. Then we compute

$$f(x) := (x-a)(x-b)g(x)$$

and ask thc oracle about a root of $f(x)$. Let the oracle's answer be c. If c is different from a and b and lies in the same row or column than a or b, we can factor n. If not, we repeat this process again.

Now let d be the minimal degree of a polynomial for which the oracle works. In this case we may choose as $g(x)$ a random polynomial of degree $d-2$. Then $f(x)$ has at most d^2 roots. The probability that n can be factored by applying the process described above once is larger or equal to

$$\frac{4d-6}{d^2} = \text{const.},$$

so by repeating the process roughly $\lceil \frac{d^2}{4d-6} \rceil$ times n will be factored. □

This result is a generalization of the problem of computing square roots in $\mathbb{Z}_n$. In [Sha93] Shamir generalized this problem in terms of factorizations of polynomials: If one can factor the polynomial $x^2 - a \equiv (x-b)(x-c) \in \mathbb{Z}_n[x]$, then b and $c = -b$ are square roots of a. To factor n, one has to find two different factorizations of this form.

Proof of Theorem 3. For fixed d and q, we may represent the number of polynomials from $\mathrm{GF}(q)[x]$ of degree d that have no root over $\mathrm{GF}(q)$ as a polynomial from $\mathbb{R}[x]$ of degree d in the variable q. Since we are only interested in asymptotic results, we consider only the coefficient of q^d.

The number of polynomials of degree d having a multiple root in $\mathrm{GF}(q)$ can be expressed as a real polynomial of degree $d-1$. Therefore in our analysis we can ignore all effects coming from multiple roots, and we may assume that all the roots of the polynomials we are considering are different.

Now let d be fixed. Let the number of monic polynomials of degree d over $\mathrm{GF}(q)$ which have exactly m roots be asymptotically equal to

$$A_m(q) = a_m q^d = \frac{c_{d-m}}{m!} q^d.$$

(The right hand side term being the definition of c_m.)

We now count the (k+1)-tuples $(f_1, f_2, \ldots, f_k, g)$, where the f_i are monic linear polynomials and g is a monic polynomial of degree $d-k$ over $\mathrm{GF}(q)$. The number of these tuples equals $q^k \cdot q^{d-k} = q^d$.

We can also count differently: If we associate to each tuple $(f_1, f_2, \ldots, f_k, g)$ the polynomial $f_1 f_2 \cdots f_k g$, then each polynomial with exactly $m \geq k$ (different) roots occurs exactly $\binom{m}{m-k} \cdot k!$ times. (Here we ignore multiple roots.)

Together we get

$$q^d = \sum_{m=k}^{d} \binom{m}{m-k} \cdot k! \cdot A_m(q).$$

Division by q^d gives:

$$1 = \sum_{m=k}^{d} \frac{m!}{(m-k)!} \cdot a_m = \sum_{m=k}^{d} \frac{c_{d-m}}{(m-k)!}.$$

With $j = d - k$ and $i = d - m$ it follows that

$$c_j = 1 - \sum_{i=0}^{j-1} \frac{c_i}{(j-i)!}.$$

This is a recursion formula for c_j. With induction we get:

$$c_k = \sum_{i=0}^{k} (-1)^i \cdot \frac{1}{i!}.$$

(It is sufficient to show that $\sum_{k=0}^{m} \left(\sum_{i=0}^{k} (-1)^i \cdot \frac{1}{i!} \right) \cdot \frac{1}{(m-k)!} = 1$. With the substitution $j = m - k + i$ this is equivalent to $\sum_{j=0}^{m} \left(\sum_{i=0}^{j} \frac{(-1)^i}{i!(j-i)!} \right) = 1$, which holds in view of the binomial theorem.)

Now we can calculate asymptotically the number of polynomials of degree d over GF(q) which have no roots:

$$A_0(q) = c_d \cdot q^d = \sum_{i=0}^{d} (-1)^i \cdot \frac{1}{i!} \cdot q^d.$$

□

Proof of Corollary 5. Part (1) is an immediate consequence of Cor. 4.

(2) Note that the mapping $\phi : \mathbb{Z}_n[x] \to \mathbb{Z}_p[x] \times \mathbb{Z}_q[x]$ defined by $\phi(f) := (f \bmod p, f \bmod q)$ is a bijection by the Chinese remainder theorem. The polynomial $f \in \mathbb{Z}_n[x]$ has at least one root if and only if both $f \bmod p$ and $f \bmod q$ have at least one root. Hence (2) follows from (1).

(3) A polynomial $f \in \mathbb{Z}_n[x]$ has exactly one root if and only if both $f \bmod p$ and $f \bmod q$ have exactly one root. Hence the fraction of polynomials with this property is approximately equal to $(e^{-1})^2$, and so the fraction of polynomials with at least two roots can be approximated by $(1-e^{-1})^2 - e^{-2} = 1 - 2e^{-1}$. □