On Error Correction in the Exponent

Chris Peikert*
MIT CSAIL

April 15, 2005

Abstract

Given a corrupted word $\mathbf{w} = (w_1, \ldots, w_n)$ from a Reed-Solomon code of distance d, there are many ways to efficiently find and correct its errors [1, 21, 10]. But what if we are instead given $(g^{w_1}, \ldots, g^{w_n})$ for some generator g of a large cyclic group — can we still correct the errors? This problem is called *error correction in the exponent*, and though it arises naturally in many areas of cryptography, it has received scant attention.

We first show that *unique decoding* in the exponent, when the number of errors $e < d/2$, is no harder than the computational Diffie-Hellman (CDH) problem in the same group. The remainder of our results are negative:

- Under mild assumptions on the parameters, we show that *bounded-distance decoding* in the exponent, under $e = d - c$ errors for any constant c, is as hard as the discrete logarithm problem in the same group.
- For *generic* algorithms that only perform group operations in a black-box manner, we show lower bounds for decoding that exactly match known algorithms.

Our generic lower bounds also extend to decisional variants of the decoding problem, and to groups in which the decisional Diffie-Hellman (DDH) problem is easy. This suggests that hardness of decoding in the exponent is a qualitatively new assumption that lies "between" the DDH and CDH assumptions, and may serve as a basis for new kinds of cryptographic schemes or improved constructions of old ones.

1 Introduction

Reed-Solomon codes and cryptography. The Reed-Solomon (RS) family of error-correcting codes [17] has proven incredibly useful throughout several areas of theoretical computer science and in many real-world applications. They are incredibly simple to define: for any positive $q \geq n \geq k$ and any evaluation set of n distinct points $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\}$ in the field $\mathbb{F}_q$, the code $\mathbb{RS}_q(\mathcal{E}, k)$ is the set of all vectors $(p(\alpha_1), \ldots, p(\alpha_n))$ where $p(x) \in \mathbb{F}_q[x]$, $\deg(p) < k$.

In addition to their elegance and many notable combinatorial properties, Reed-Solomon codes also admit efficient algorithms for correcting errors in codewords: the Berlekamp-Welch algorithm [1] for correcting errors up to half the distance of the code, and the groundbreaking list-decoding algorithms of Sudan [21] and Guruswami and Sudan [10], to name a few.

Reed-Solomon codes also play a fundamental role in modern cryptography. McEliece and Sarwate first observed [13] that in Shamir's secret-sharing scheme [19], the sharing process can be seen

*32 Vassar St, Cambridge, MA 02139, cpeikert@mit.edu

as encoding the secret under an RS code, and reconstructing the secret in the presence of malicious players is essentially decoding under errors and/or erasures. Indeed, the shares in Shamir's scheme are simply the symbols of an RS codeword, which are the values of a low-degree polynomial at many distinct points. Shamir's scheme lies at the foundation of general multiparty computation, threshold cryptography, and more.

Another reason that Shamir secret sharing/RS encoding is so useful to cryptography is that polynomials can be efficiently "interpolated in the exponent:"[1] given k values of the form $g^{p(\alpha_i)}$ for a polynomial p of degree less than k, it is possible to compute $g^{p(x)} = \prod_i (g^{p(\alpha_i)})^{\lambda_i}$ for any x and appropriate Lagrange coefficients λ_i. For this reason, cryptographic protocols relying on the hardness of computing discrete logs can often be made secure against a "halting" adversary (i.e., one that may withhold its shares, but never mis-reports them), while remaining very efficient.

Introducing errors in the exponent. A *malicious* adversary, on the other hand, may lie about its shares, which introduces errors instead of erasures. This motivates us to consider the problem of *correcting errors* "in the exponent." That is, we want to correct up to e errors in words of the form $(g^{w_1}, \ldots, g^{w_n})$ where $\mathbf{w} = (w_1, \ldots, w_n)$ is an RS codeword, and G is an arbitrary cyclic group of known order q that is generated by g. In general, Reed-Solomon codes can be defined over any field $\mathbb{F}$ — but in order for this problem to make sense, the symbols of $\mathbb{F}$ must be interpreted as residues modulo q. Therefore we will take $\mathbb{F}$ to be $\mathbb{Z}_q$ for any prime q. The cyclic group G may be any group of order q; for example, the subgroup of $\mathbb{Z}_p$ where $p = \mu q + 1$ for some integer μ, or a group defined over an elliptic curve.

1.1 Applications

While error correction in the exponent is a very interesting problem in its own right, it is also heavily motivated by existing work.

In the positive direction, an error correction algorithm would be highly desirable, because it would lead to improvements in robustness (i.e., correctness in the presence of cheating players) and efficiency of many multiparty cryptographic protocols. Currently, these protocols often require either expensive zero-knowledge proofs of correct operation, or more clever (but still somewhat expensive) tricks using verifiable secret sharing schemes. In any case, these steps cost extra rounds of communication and computation, which could avoided by instead having the parties perform local error correction (with the beneficial side-effect of identifying cheating parties).

Protocols in the literature which would benefit from error correction in the exponent include, but are certainly not limited to: threshold DSS key generation and signature protocols [9], related threshold ElGamal protocols [15], protocols for multiplication of shared secrets in the exponent [16], distributed pseudorandom generators, functions, and verifiable random functions [14, 8, 5], and many others.

In the negative direction, hardness of error correction is a novel assumption that may provide a foundation for new kinds of cryptographic schemes, or improved constructions for existing primitives.

[1] Using the language of coding theory, we might describe this operation as "erasure decoding in the exponent."

1.2 Our Results

We consider the problem of correcting errors in the exponent for the code $\mathbb{RS}_q(\mathcal{E}, k)$, defined over the field $\mathbb{Z}_q$.

First, we observe that *unique decoding* in the exponent, when the number of errors e is less than half the distance of the code, is no harder than the (computational) Diffie-Hellman (CDH) problem [7] in the same group. The remainder of our results are negative:

- Under mild assumptions on the parameters, we show that *bounded-distance decoding* in the exponent, under $e = n - k - c$ errors for any constant c, is as hard as the discrete logarithm problem in the same group.
- For *generic* algorithms (as defined by Shoup [20]) that only perform group operations in a black-box manner, we show lower bounds for decoding that exactly match known algorithms.

Our generic lower bounds also extend to decisional variants of the decoding problem, and to groups in which the decisional Diffie-Hellman (DDH) problem is easy. This suggests that hardness of decoding in the exponent is a qualitatively new assumption that lies "between" the DDH and CDH assumptions, and may serve as a basis for new kinds of cryptographic schemes or improved constructions of old ones.

1.3 Related Work

We are aware of only one work which directly addresses error correction in the exponent: Canetti and Goldwasser [3] gave a simple, efficient decoding algorithm which works when e and k are $O(\sqrt{n})$. (See Proposition 2.1 for a generalization.) This provides an inexpensive way to achieve mild robustness in their threshold version of the Cramer-Shoup cryptosystem [6]. They also pose the general case as an interesting open problem.

A few recent works have investigated the hardness of various "plain" (i.e., not in the exponent) decoding tasks for Reed-Solomon codes. Cheng and Wan [4], somewhat surprisingly, showed that (under an appropriate number of errors) certain list- and bounded-distance decoding problems are as hard as computing discrete logs. However, their setting differs from ours in important ways: for them, q is small (polynomial in n), and they relate decoding to the discrete log problem in the field $\mathbb{F}_{q^h}$ for a somewhat large h. In contrast, we are concerned with discrete log-related problems in groups of order q, where q is exponentially large.

Guruswami and Vardy [11] resolved a long-standing open problem, showing that maximum-likelihood decoding (i.e., finding the nearest codeword) of Reed-Solomon codes is NP-hard. More specifically, they showed that it is hard to distinguish whether a word is at distance $n - k$ or $n - k - 1$ from a Reed-Solomon code. Of course, the problem remains NP-hard when placed "in the exponent." However, their results are also incomparable to ours: they show a stronger form of hardness, but only in the worst case, for a very large number of errors, and for a carefully-crafted evaluation set $\mathcal{E}$. In contrast, we show weaker forms of hardness, but in the average case, under fewer errors, and for any $\mathcal{E}$. We again stress, however, that both works [4, 11] are concerned only with plain decoding.

Notation. We denote a vector $\mathbf{x}$ in boldface and its value at index i by x_i. For two vectors $\mathbf{x}, \mathbf{y}$ of the same length, define $\Delta(\mathbf{x}, \mathbf{y})$ to be the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, i.e. the

number of indices i for which $x_i \neq y_i$. Define $\mathrm{wt}(\mathbf{x}) = \Delta(0, \mathbf{x})$. For a code $\mathbb{C}$ and a vector $\mathbf{x}$, define $\Delta(\mathbf{x}, \mathbb{C}) = \min_{\mathbf{y} \in \mathbb{C}} \Delta(\mathbf{x}, \mathbf{y})$. Finally, denote $\{1, \ldots, n\}$ by $[n]$.

2 Initial Observations and Upper Bounds

Decoding with a Diffie-Hellman oracle. Clearly, unique decoding in the exponent under $e < (n-k+1)/2$ errors is no harder than the discrete log problem: given $(g^{x_1}, \ldots, g^{x_n})$, taking discrete logs yields $(x_1, \ldots, x_n)$, which can be corrected using the standard algorithms [1]. However, this approach is actually overkill: it is, in fact, enough to have an oracle for the (computational) Diffie-Hellman problem in G. This is because all the operations of the Berlekamp-Welch algorithm can be performed efficiently, except for multiplication and inversion in the exponent. Multiplication is immediately provided by the Diffie-Hellman oracle. Inversion can be implemented as follows: on input g^a, compute $g^{a^{-1} \bmod q} = g^{a^{q-1} \bmod q}$ by the method of repeated squaring, invoking the oracle on g^x, g^x to get $g^{x^2 \bmod q}$. (We remark that this approach requires that q be known.)

Decoding by enumeration. Another approach to unique decoding is to merely enumerate over all subsets of size k of received shares. For each subset K, do Lagrange interpolation in the exponent (over K) to each point in $\mathcal{E}$, counting the number of disagreements with the received shares. It is easy to show that when the number of disagreements is at most e, the shares in K are all correct, and the entire codeword can be recovered from them. Unfortunately, this approach takes time $\binom{n}{k}$, which is not $\mathrm{poly}(n)$ in general.

A similar, but more efficient randomized approach was given in [3] for the case $e+1 = k = O(\sqrt{n})$. Here we generalize it to arbitrary e,k:

Proposition 2.1. *For any $e, k < n$ such that $e < (n-k+1)/2$, there is an algorithm for unique decoding in the exponent which performs $O\left(nk(\log q) \cdot \binom{n}{k} / \binom{n-e}{k}\right)$ group operations and succeeds with all but negligible (in n) probability. When $ek = O(n \log n)$, the algorithm performs poly$(n) \cdot O(\log q)$ group operations.*

Proof. The algorithm works exactly as the enumeration algorithm, except with an independent, random set K for each iteration, for a suitable maximum number of iterations.

Correctness of the algorithm immediately follows from the distance property of $\mathbb{RS}_q(\mathcal{E}, k)$. We now analyze the runtime: each iteration can be done with $O(nk \log q)$ group operations, using repeated squaring to raise each share to its Lagrange coefficient. A given iteration succeeds iff all k of the chosen shares are correct, and the probability of this event is:

$$\frac{\binom{n-e}{k}}{\binom{n}{k}} = \frac{(n-e)!(n-k)!}{(n)!(n-e-k)!}.$$

There are two ways to bound this quantity from below: we can write $\frac{(n-e)!}{n!} \geq n^{-e}$ and $\frac{(n-k)!}{(n-e-k)!} \geq (n-e-k)^e$, *or* we can write $\frac{(n-k)!}{n!} \geq n^{-k}$ and $\frac{(n-e)!}{(n-e-k)!} \geq (n-e-k)^k$. Taking the best of the two, we get a bound of:

$$\left(1 - \frac{e+k}{n}\right)^{\min(e,k)} = \exp(-O(\min(e,k)(e+k)/n)) = \exp(-O(ek/n)) = 1/\mathrm{poly}(n).$$

Therefore the algorithm can be made to run in $\mathrm{poly}(n)$ time and succeed with high probability. □

Taking the best of all the above approaches, we see that the complexity of unique decoding in the exponent is upper-bounded by the complexity of the CDH problem and by $\binom{n}{k}/\binom{n-e}{k}$. The remainder of this paper will be devoted to establishing lower bounds.

3 Bounded-Distance Decoding in the Exponent

In this section, we show that *bounded-distance decoding* (a relaxation of unique decoding) in the exponent is as hard as the discrete log problem, under a large number of errors. For convenience, we define the following code for a generator g of a cyclic group G of order q: $\mathbb{C}_q(\mathcal{E}, k, g) = \{(g^{w_1}, \ldots, g^{w_n}) \ : \ \mathbf{w} \in \mathbb{RS}_q(\mathcal{E}, k)\}$.

Problem: Bounded-distance decoding of $\mathbb{C}_q(\mathcal{E}, k, g)$ under e errors. We denote this problem by $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,e}$.

Instance: A generator g of G, and a word $\mathbf{x}$ such that $\Delta(\mathbf{x}, \mathbb{C}_q(\mathcal{E}, k, g)) \leq e$.

Output: Any codeword $\mathbf{p} \in \mathbb{C}_q(\mathcal{E}, k, g)$ such that $\Delta(\mathbf{p}, \mathbf{x}) \leq e$.

We will relate BDDE-RS to the following problem:

Problem: Finding a nontrivial representation of the identity element $1 \in G$, with respect to a uniform base of n elements. We denote this problem by $\mathsf{FIND\text{-}REP}_n$.

Instance: A base $(x_1, \ldots, x_n) \in G^n$, chosen uniformly.

Output: Any nontrivial $(\lambda_1, \ldots, \lambda_n) \in \mathbb{Z}_q^n$ such that $\prod_{i=1}^n x_i^{\lambda_i} = 1$.

Boneh and Golle [2] showed that $\mathsf{FIND\text{-}REP}_n$ is as hard as computing discrete logs. We briefly summarize their proof here: given an algorithm $\mathcal{B}$ that solves $\mathsf{FIND\text{-}REP}_n$ in G, we construct an algorithm to solve the discrete log problem in G. On input (g, y) where $\log_g y$ is desired, choose $(r_1, \ldots, r_n)$ and $(s_1, \ldots, s_n)$ from $\mathbb{Z}_q^n$ independently at random, and let $x_i = g^{r_i} y^{s_i}$. Run $\mathcal{B}$ on $(x_1, \ldots, x_n)$ and get output $(\lambda_1, \ldots, \lambda_n)$. If $\sum s_i\lambda_i \neq 0 \bmod q$, output $-\frac{\sum r_i\lambda_i}{\sum s_i\lambda_i} \bmod q$.

The analysis is straightforward: first observe that the constructed $(x_1, \ldots, x_n)$ is uniform over G^n. Furthermore, all r_i and s_i are information-theoretically hidden from $\mathcal{B}$. Therefore if $(\lambda_1, \ldots, \lambda_n)$ is nontrivial, $\Pr[\sum s_i\lambda_i = 0 \bmod q] = 1/q$, which is negligible. Finally, suppose $z = \log_g y$. Then $1 = \prod x_i^{\lambda_i} = \prod g^{\lambda_i(r_i + zs_i)}$, which implies $\sum \lambda_i(r_i + zs_i) = 0 \bmod q$. Solving for z, we see that the algorithm's output is correct.

Now, in order to reduce FIND-REP to BDDE-RS, we will use the following technical lemma:

Lemma 3.1. *For any positive integer $c \leq n - k$, for a random $\mathbf{x} \in G^n$,*

$$\Pr_{\mathbf{x}}[\Delta(\mathbf{x}, \mathbb{C}_q(\mathcal{E}, k, g)) > n - k - c] \quad \leq \quad \frac{q^c \cdot n^{2c}}{\binom{n}{k+c}}.$$

Proof of Lemma 3.1. It is apparent that $\Delta(\mathbf{x}, \mathbb{C}_q(\mathcal{E}, k, g)) \leq n - k - c$ if (and only if) there exists some set of indices $S \subseteq [n]$, $|S| = k + c$, satisfying the following "low degree" condition: the points $\{(\alpha_i, \log_g x_i)\}_{i \in S}$ lie on a polynomial of degree less than k. Define $\mathcal{S} = \{S \subseteq [n] \ : \ |S| = k + c\}$. For every $S \in \mathcal{S}$, define X_S to be the 0-1 random variable indicating whether S satisfies the low degree condition, taken over the random choice of $\mathbf{x}$. Let $X = \sum_{S \in \mathcal{S}} X_S$.

It is clear that for all $S \in \mathcal{S}$, $\Pr[X_S = 1] = q^{-c}$, so $E[X] = \binom{n}{k+c}/q^c$. Also,

$$\Pr[\Delta(\mathbf{x}, \mathbb{C}_q(\mathcal{E}, k, g)) > n - k - c] = \Pr[X = 0] \leq \Pr\left[|X - E[X]| \geq E[X]\right] \leq \frac{\sigma_X^2}{E[X]^2}$$

by Chebyshev's inequality, where σ_Z^2 denotes the variance of a random variable Z.

It remains to analyze $\sigma_X^2 = E[X^2] - E[X]^2$. The central observation is that for a large fraction of $S, S' \in \mathcal{S}$, $X_S, X_{S'}$ are pairwise independent. In particular, for any S, S' such that $|S \cap S'| \leq k$, X_S and $X_{S'}$ are pairwise independent, hence $E[X_S X_{S'}] = E[X_S]E[X_{S'}]$. And for all other $S \neq S'$, $E[X_S X_{S'}] \leq 1/q^c$. The number of pairs of distinct $S, S' \in \mathcal{S}$ such that $|S \cap S'| > k$ is at most $\binom{n}{k+c}\binom{k+c}{k+1}\binom{n-k-1}{c-1}$. Also, we note that for any indicator variable Z, $\sigma_Z^2 \leq E[Z]$. Putting these observations together, we obtain the following bound on σ_X^2:

$$\begin{aligned}
\sigma_X^2 &= \sum_{S \in \mathcal{S}} \sigma_{X_S}^2 + \sum_{\substack{S, S' \in \mathcal{S} \\ S \neq S'}} \left(E[X_S X_{S'}] - E[X_S]E[X_{S'}]\right) \\
&\leq \frac{\binom{n}{k+c}}{q^c} + \sum_{\substack{S, S' \in \mathcal{S} \\ |S \cap S'| > k}} E[X_S X_{S'}] \quad \leq \quad E[X]\left[1 + \binom{k+c}{c+1}\binom{n-k-1}{c-1}\right].
\end{aligned}$$

Since $k + c \leq n$, we may twice apply the (very naive) bound of $\binom{n}{y} \leq n^y$, and the claim follows. □

Theorem 3.2. *For any positive integer c, and for any n, k, and q such that $\binom{n}{k+c} \geq 2q^c n^{2c}$, if $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$ is solvable in random time n^d for some constant d, then the discrete log problem in G is solvable in random time $O(n^d)$.*

Remark 3.3. For parameters that typically occur in cryptography, the hypothesis of Theorem 3.2 is quite weak. For instance, suppose $n = \log^2 q$ and $k = \Theta(n)$: then $\binom{n}{k+c} \approx 2^{\Theta(n)}$, which is super-polynomial in q, hence certainly large enough to apply the Theorem.

Proof of Theorem 3.2. Suppose that algorithm $\mathcal{D}$ solves $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$. It will suffice to construct an algorithm $\mathcal{A}$ that solves $\mathsf{FIND\text{-}REP}_n$ in G. (We note that $\mathsf{FIND\text{-}REP}_n$ is random self-reducible, therefore using standard amplification techniques it is sufficient to solve $\mathsf{FIND\text{-}REP}_n$ with probability non-negligible in n.)

$\mathcal{A}$ works as follows: on input $\mathbf{x} = (x_1, \ldots, x_n)$, where $\mathbf{x}$ is uniform over G^n, immediately run $\mathcal{D}(g, \mathbf{x})$. If $(g, \mathbf{x})$ is actually an instance of $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$, $\mathcal{D}$ will output some $\mathbf{p} = (p_1, \ldots, p_n)$ where $\Delta(\mathbf{p}, \mathbf{x}) \leq n - k - c$. Take any $k+1$ indices $E \subseteq [n]$ such that $x_i = p_i$ for $i \in E$. By Lagrange interpolation, compute non-trivial λ_i for all $i \in E$ such that $\prod_{i \in E} x_i^{\lambda_i} = 1$. Let $\lambda_i = 0$ for all $i \notin E$, and output $(\lambda_1, \ldots, \lambda_n)$.

It remains to bound the probability that $(g, \mathbf{x})$ is an instance of $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$, i.e. that $\Delta(\mathbf{x}, \mathbb{C}_q(\mathcal{E}, k, g)) \leq n - k - c$. By Lemma 3.1, we immediately see that this probability is at least $1/2$. □

4 Generic Algorithms for Noisy Polynomial Interpolation

Generic algorithms. Shoup [20] proposed the *generic algorithms* framework for problems relating to arbitrary groups, and proved several lower bounds for computing discrete logs and related

problems. Informally, a generic algorithm only performs group operations in a black-box manner; it does not use any particular property of the *representation* of group elements.

Formally, we consider a group G, an arbitrary set $S \subset \{0,1\}^*$ with $|S| \geq |G|$, and a random injective *encoding function* $\sigma : G \to S$. We are only concerned with cyclic groups G of prime order q, independent of their representation. Such group are all isomorphic to $\mathbb{Z}_q$ under addition, so we will assume without loss of generality that $G = \mathbb{Z}_q$ under group operation $+$.

A generic algorithm $\mathcal{A}$ has access to an *encoding list* $(\sigma(x_1), \ldots, \sigma(x_t))$ of elements $x_1, \ldots, x_t \in \mathbb{Z}_q$. $\mathcal{A}$ can make unit-time queries of the form $x_i \pm x_j$ to a *group oracle* by specifying the operation and the indices i, j into the encoding list; the answer $\sigma(x_{t+1})$, where $x_{t+1} = x_i \pm x_j$, is appended to the list. The query complexity of a generic algorithm is the number of elements in its encoding list (including any provided as input) when it terminates.

The probability space of an execution of $\mathcal{A}$ consists of the random choice of input, the random function σ, and the coins of $\mathcal{A}$. If we bound the success probability of $\mathcal{A}$ over this space, then it follows that for *some* encoding function σ, the same bound applies when the probability is taken only over the input and $\mathcal{A}$'s coins. Therefore any algorithm which uses the group in a "black-box" manner is subject to the bound.

We remark that most general-purpose algorithms for discrete log and related problems are indeed generic. One exception is the index calculus method, which requires a notion of "smoothness" in the group G. Thus far, index calculus methods have not been successfully applied to groups over the kinds of elliptic curves that are typically used in cryptography.

Noisy polynomial interpolation. We now consider a problem which we call "noisy polynomial interpolation," which is closely related to decoding for Reed-Solomon codes. (See Remark 4.1 below for details on this relationship.) This is exactly the problem which tends to appear in many multiparty cryptographic protocols.

> **Problem:** Generic noisy polynomial interpolation at a fixed point $\alpha_0 \notin \mathcal{E}$ under $e < (n-k+1)/2$ errors. We denote this problem by $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$.
>
> **Instance:** An initial encoding list $(\sigma(P(\alpha_1)+e_1), \ldots, \sigma(P(\alpha_n)+e_n), \sigma(1))$ for a random $P(x) \in \mathbb{Z}_q[x]$, $\deg(P) < k$, and a random $\mathbf{e} \in \mathbb{Z}_q^n$ such that $\mathrm{wt}(\mathbf{e}) = e$.
>
> **Output:** $\sigma(P(\alpha_0))$.

Remark 4.1. GNPI is potentially a *strictly easier* problem than full decoding: it could be the case that interpolating a noisy polynomial at some specific, rare point α_0 is easier than recovering the entire codeword (i.e., interpolating at all points $\alpha_1, \ldots, \alpha_n$). Conversely, recovering the entire codeword would permit generic Lagrange interpolation of the polynomial at *any* point α_0. Therefore, the bound for GNPI provided by Theorem 4.2 is potentially stronger than one which might be provided for the full-decoding task.

Theorem 4.2. *A generic algorithm for $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$ making m queries succeeds with probability at most $(m+1)^2 \left(1/q + \binom{n-k}{e}/\binom{n}{e}\right)$.*

Corollary 4.3. *If $ek = \omega(n \log n)$ and q is super-polynomial in n, no generic algorithm that runs in time poly(n) solves $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$, except with probability negligible in n. In particular, the algorithm of Canetti and Goldwasser [3] (described in Section 2) is optimal.*

Proof of Corollary 4.3. First, $\binom{n-k}{e}/\binom{n}{e} \leq \left(\frac{n-k}{n}\right)^e = (1-k/n)^e = \exp(-\Omega(ek/n))$, which is negligible in n. Since $1/q$ is negligible as well, the total success probability is negligible. □

Proof of Theorem 4.2. We can write the real interaction between a generic algorithm $\mathcal{A}$ and its oracle as a game, which proceeds as follows: let $P_0, \ldots, P_{k-1}$ and $E_1, \ldots, E_n$ be indeterminants. First, the game chooses $\mathbf{p} = (p_0, \ldots, p_{k-1}) \leftarrow \mathbb{Z}_q^k$ and $\mathbf{e} \in \mathbb{Z}_q^n$ uniformly, such that $\mathrm{wt}(\mathbf{e}) = e$. While interacting with $\mathcal{A}$, the game will maintain a list of linear polynomials $F_1, \ldots, F_t \in \mathbb{Z}_q[P_0, \ldots, P_{k-1}, E_1, \ldots, E_n]$. Concurrently, $\mathcal{A}$ will have an encoding list $(\sigma(x_1), \ldots, \sigma(x_t))$ where $x_j = F_j(\mathbf{p}, \mathbf{e})$. Furthermore, the game defines an "output polynomial" F_0, which corresponds to the correct output.

Initially, $t = n+1$, $F_j = E_j + \sum_{i=0}^{k-1} P_i \alpha_j^i$ for $j \in [n]$, and $F_{n+1} = 1$. The output polynomial is $F_0 = \sum_{i=0}^{k-1} P_i \alpha_0^i$.

Whenever $\mathcal{A}$ makes a query for $x_i \pm x_j$, the game computes $F_{t+1} = F_i \pm F_j, x_{t+1} = F_{t+1}(\mathbf{p}, \mathbf{e}), \sigma_{t+1} = \sigma(x_{t+1})$, and appends σ_{t+1} to $\mathcal{A}$'s encoding list. When $\mathcal{A}$ terminates, we may assume that it always outputs some σ_j it received from the oracle (otherwise $\mathcal{A}$ only succeeds with probability at most $\frac{1}{q-m}$). Then $\mathcal{A}$ succeeds iff $\sigma_j = \sigma(F_0(\mathbf{p}, \mathbf{e}))$.

The ideal game. We now consider an "ideal game" between $\mathcal{A}$ and a different oracle, in which each *distinct polynomial* F_j is mapped to a *distinct*, random σ_j, independent of the value $F_j(\mathbf{p}, \mathbf{e})$. More formally, the game proceeds as follows: initially, $(\sigma_1, \ldots, \sigma_{n+1})$ is just a list of distinct random elements of S corresponding to polynomials $F_1, \ldots, F_{n+1}$ defined above. Whenever $\mathcal{A}$ asks for $x_i \pm x_j$ as its $(t+1)$st query, the game computes $F_{t+1} = F_i \pm F_j$. If $F_{t+1} = F_\ell$ for any $\ell \leq t$, the game sets $\sigma_{t+1} = \sigma_\ell$, otherwise it chooses σ_{t+1} to be a random element of $S - \{\sigma_1, \ldots, \sigma_t\}$. Finally, when $\mathcal{A}$ terminates, the game chooses a random value σ_0 from $S - \{\sigma_1, \ldots, \sigma_m\}$, corresponding to F_0. $\mathcal{A}$ succeeds in this game if it outputs σ_0; since $\mathcal{A}$ only produces output from $\{\sigma_1, \ldots, \sigma_m\}$, the success probability in the ideal game is zero.

It is easy to see that $\mathcal{A}$'s success probability in the real game is identical to its success probability in the ideal game, *conditioned* on a "failure event" $\mathcal{F}$ not occurring. The event $\mathcal{F}$ is that $F_i(\mathbf{p}, \mathbf{e}) = F_{i'}(\mathbf{p}, \mathbf{e})$ for some $F_i \neq F_{i'}$, where $i, i' \in \{0, \ldots, m\}$, and the probability is taken over $\mathbf{p}, \mathbf{e}$.

Analysis of the games. We now analyze $\Pr[\mathcal{F}]$: for any $F_i \neq F_{i'}$, consider $F = (F_i - F_{i'}) \in \mathbb{Z}_q[P_0, \ldots, P_{k-1}, E_1, \ldots, E_n]$. Suppose that in $\mathbf{e}$, the values e_j for indices $j \in M = \{m_1, \ldots, m_e\}$ are chosen uniformly, while the others are zero. Then we can consider a polynomial F' in the indeterminants $P_0, \ldots, P_{k-1}$ and $E_{m_1}, \ldots, E_{m_e}$, where F' is simply F with zero substituted for each E_j, $j \notin M$.

Let $\mathbf{e}' = (\mathbf{e}_{m_1}, \ldots, \mathbf{e}_{m_e})$. We are then interested in $Pr_{\mathbf{p},\mathbf{e}'}[F'(\mathbf{p}, \mathbf{e}') = 0]$. There are two cases: if F' is nontrivial, then this probability is $1/q$ by Schwartz's lemma [18], because $\mathbf{p}$ and $\mathbf{e}'$ are chosen uniformly. Therefore it remains to bound $\Pr_{\mathbf{p},\mathbf{e}}[F' = 0]$.

In order to have $F' = 0$, the constant term and all the coefficients of P_ℓ must be zero in F', and hence also in F. By its construction, F is a nontrivial linear combination of $F_0, \ldots, F_n$, and $F_{n+1} = 1$: i.e., there exist $\mathbf{c} = (c_0, \ldots, c_n) \in \mathbb{Z}_q^{n+1}$ and $d \in \mathbb{Z}_q$ such that

$$F \quad = \quad d + \sum_{j=0}^{n} c_j F_j \quad = \quad d + \sum_{j=1}^{n} c_j E_j + \sum_{\ell=0}^{k-1} P_\ell \cdot \sum_{j=0}^{n} c_j \alpha_j^\ell.$$

Therefore we have $d = 0$ and $A\mathbf{c} = 0$, where A is a Vandermonde matrix with $A_{\ell+1,j+1} = \alpha_j^\ell$ for $j = 0, \ldots, n$ and $\ell = 0, \ldots, k-1$. Because any k columns of A are linearly independent and F is nontrivial, we have $\mathrm{wt}(\mathbf{c}) \geq k+1$. In order for $F' = 0$, it must be that $c_j = 0$ for every $j \in M$. Because the set M is chosen independently of $\mathbf{c}$, the probability of this event is at most $\binom{n-k}{e}/\binom{n}{e}$. Finally, by a union bound over all pairs $F_i \neq F_{i'}$, we obtain the result. □

4.1 Relation to the DDH Problem

In this section, we show evidence that the noisy polynomial interpolation problem in G is not as easy as the Decisional Diffie-Hellman (DDH) problem in G. Specifically, for the GNPI problem, we show lower bounds for generic algorithms that are augmented with a DDH oracle.

Such lower bounds imply that, even in groups in which the DDH problem is easy, noisy polynomial interpolation may still be hard. Such a scenario is not just idle speculation: there are reasonable instances of so-called "gap Diffie-Hellman" groups [12], in which the DDH problem is *known* to be easy, but the *computational* Diffie-Hellman problem is believed to be hard. Recalling from Section 2 that GNPI is no harder than the CDH problem, this suggests that GNPI may be a problem of intermediate hardness, located strictly between the (easy) DDH problem and the (assumed hard) CDH problem.

Augmented generic algorithms. We augment a generic algorithm $\mathcal{A}$ with a DDH oracle as follows: at any time, $\mathcal{A}$ can submit to the DDH oracle a triple (a, b, z) of indices into its encoding list. The oracle replies whether $x_a \cdot x_b = x_z \bmod q$.

Theorem 4.4. *A generic algorithm, augmented with a DDH oracle, for $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$ making m_G queries to its group oracle and m_D queries to its DDH oracle succeeds with probability at most $\left((m_G+1)^2 + 2m_D\right)\left(1/q + \binom{n-k}{e}/\binom{n}{e}\right)$.*

Corollary 4.5. *If $ek = \omega(n \log n)$ and q is super-polynomial in n, no generic algorithm augmented with a DDH oracle that runs in time poly(n) solves $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$, except with probability negligible in n.*

Proof Sketch of Theorem 4.4. As in the proof of Theorem 4.2, we consider "real" and "ideal" games, and bound the probability of a failure event.

Both games proceed much in the same way: they maintain a list of polynomials F_i and answer queries to the group oracle as before. The games answer DDH queries (a, b, z) in the following way: In the real game, respond "yes" if $F_a(\mathbf{p}, \mathbf{e}) \cdot F_b(\mathbf{p}, \mathbf{e}) = F_z(\mathbf{p}, \mathbf{e})$, where the multiplication is done in $\mathbb{Z}_q$. In the ideal game, respond "yes" if $F_a \cdot F_b = F_z$, where the multiplication is of formal polynomials in $\mathbb{Z}_q[P_0, \ldots, P_{k-1}, E_1, \ldots, E_n]$.

The failure event $\mathcal{F}$ is the union of the old failure event (from the proof of Theorem 4.2) with the event that, for some query (a, b, z) to the DDH oracle, $F_a(\mathbf{p}, \mathbf{e}) \cdot F_b(\mathbf{p}, \mathbf{e}) - F_z(\mathbf{p}, \mathbf{e}) = 0$ when $F_a \cdot F_b - F_z \neq 0$.

As before, suppose $M = \{m_1, \ldots, m_e\}$ is the set of indices such that $\{e_j\}_{j \in M}$ are chosen uniformly, while the others are zero, and let $\mathbf{e}' = (e_{m_1}, \ldots, e_{m_e})$. For a particular query (a, b, z) such that $F = F_a \cdot F_b - F_z \neq 0$, consider the polynomial $F' \in \mathbb{Z}_q[P_0, \ldots, P_{k-1}, E_{m_1}, \ldots, E_{m_e}]$ which is defined to be F with zero substituted for all E_j, $j \notin M$. Define F'_a, F'_b, F'_z similarly, so $F' = F'_a F'_b - F'_z$. Certainly the total degree of F' is at most 2. If $F' \neq 0$, then by Schwartz's lemma [18], $\Pr[F'(\mathbf{p}, \mathbf{e}') = 0] \leq 2/q$.

It remains to bound $\Pr_{\mathbf{p},\mathbf{e}}[F' = 0]$. In order to have $F \neq 0$ and $F' = 0$, one of two cases must be true: (1) F_a or F_b (or both) is a constant polynomial, or (2) F_a, F_b are both non-constant.

In case (1), F is nonzero, linear, and is a linear combination of $F_1, \ldots, F_{n+1}$. As argued in the proof of Theorem 4.2, $\Pr[F' = 0 \mid F \neq 0] \leq \binom{n-k}{e}/\binom{n}{e}$.

For case (2), we will need some notation: for any monomial X in a polynomial G, define $\text{coeff}_X(G)$ to be the coefficient of the X term in G. It is easy to show (by considering the quadratic terms of F') that for either $i = a$ or $i = b$, F_i' is a constant polynomial. Now as in the proof of Theorem 4.2, $\Pr[F_a' \text{ or } F_b' \text{ is constant} \mid F_a, F_b \text{ are non-constant}] \leq 2\binom{n-k}{e}/\binom{n}{e}$. Taking a union bound over all queries to the DDH oracle, we get the claimed result. □

4.2 Decisional Variants

Certain *decisional* versions of the noisy polynomial interpolation problem are also hard for generic algorithms. Here, in addition to the noisy points of the polynomial, the algorithm is given the correct value $P(\alpha_0)$ and a truly random value (in random order), and simply must decide which is which. We denote this problem by $\mathsf{DGNPI}_{q,\mathcal{E},\alpha_0,k,e}$. The hardness of DGNPI implies that $P(\alpha_0)$ "looks random," given the noisy values of the polynomial.

Theorem 4.6. *A generic algorithm for $\mathsf{DGNPI}_{q,\mathcal{E},\alpha_0,k,e}$ making m queries succeeds with probability at most $\frac{1}{2} + 2m^2\left(1/q + \binom{n-k}{e}/\binom{n}{e}\right)$.*

The proof of Theorem 4.6 is nearly identical to that of Theorem 4.2.

In fact, we can extend the definition of DGNPI instances to include the value of the polynomial P at *several* distinct points $\beta_0, \ldots, \beta_r \notin \mathcal{E}$, instead of just at α_0. These evaluations "look random" to generic algorithms, with a distinguishing advantage bounded by $2m^2\left(1/q + \binom{n-(k-r)}{e}/\binom{n}{e}\right)$. Also, as in Section 4.1, we can prove that DGNPI is hard for generic algorithms that are augmented with a DDH oracle. We defer the details to the final version.

5 Conclusions and Open Problems

We have shown evidence that error correction (of Reed-Solomon codes) in the exponent is hard, and that its hardness seems to be qualitatively different than that of the Diffie-Hellman problems. We can think of several related open problems, including:

- Is there some other family of codes which admits an efficient (preferably generic) algorithm for decoding in the exponent, and can it be used as the basis of a secret-sharing scheme?
- Assuming that decoding in the exponent is hard, can we provide new constructions of standard (or new) cryptographic primitives? Such constructions would be useful, both as a hedge against possible attacks on more commonly-used assumptions, and for any unique properties or efficiency gains they might provide.
- Is there a reduction to decoding in the exponent under fewer than $n - k - O(1)$ errors, from some well-studied problem like computing discrete logs, CDH, or DDH?

In addition, the general idea of correcting errors in "partially hidden" data seems wide open and ripe with interesting problems.

Acknowledgements

The author would like to gratefully thank Shafi Goldwasser, Ran Canetti, Alon Rosen, and Tal Rabin for helpful comments and discussions.

References

[1] E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[2] D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. In *ACM Conference on Computer and Communications Security 2002*, pages 68–77, 2002.

[3] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against chosen ciphertext attack. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592, pages 90–106. Springer-Verlag, 1999.

[4] Q. Cheng and D. Wan. On the list and bounded distance decodability of the Reed-Solomon codes. In *Proc. FOCS 2004*, pages 335–341. IEEE Computer Society, 2004.

[5] R. Cramer and I. Damgård. Secret-key zero-knowlegde and non-interactive verifiable exponentiation. In *1st TCC*, pages 223–237, 2004.

[6] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO'98*, 1998.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[8] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *6th PKC*, pages 1–17, 2003.

[9] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold dss signatures. In *Advances in Cryptology — Eurocrypt '96*, pages 354–371, 1996.

[10] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *IEEE Symposium on Foundations of Computer Science*, pages 28–39, 1998.

[11] V. Guruswami and A. Vardy. Maximum-likelihood decoding of Reed-Solomon codes is NP-hard. In *SODA*, 2005.

[12] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.

[13] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM*, 24(9):583–584, 1981.

[14] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and kdcs. In *Advances in Cryptology — Eurocrypt '99*, pages 327–346, 1999.

[15] C. Park and K. Kurosawa. New ElGamal type threshold digital signature scheme. *IEICE Trans. Fundamentals*, E79-A(1):86–93, January 1996.

[16] M. D. Raimondo and R. Gennaro. Secure multiplication of shared secrets in the exponent. Cryptology ePrint Archive, Report 2003/057, 2003.

[17] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8(2):300–304, June 1960.

[18] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[19] A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, 1979.

[20] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. Eurocrypt '97*, pages 256–266, 1997.

[21] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.