On Error Correction in the Exponent

Chris Peikert

MIT CSAIL, 32 Vassar St, Cambridge, MA, 02139.
cpeikert@theory.csail.mit.edu

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. But what if we are instead given $(g^{w_1}, \ldots, g^{w_n})$ where g generates some large cyclic group — can the errors still be corrected efficiently? This problem is called *error correction in the exponent*, and though it arises naturally in many areas of cryptography, it has received little attention.

We first show that *unique decoding* and *list decoding* in the exponent are 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 $e = d - k^{1-\epsilon}$ errors for any $\epsilon > 0$, is as hard as the discrete logarithm problem in the same group.
– For *generic* algorithms (as defined by Shoup, Eurocrypt 1997) that treat the group as a "black-box," 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.

1 Introduction

Reed-Solomon codes and cryptography. The Reed-Solomon (RS) family of error-correcting codes [19] has proven incredibly useful throughout several areas of theoretical computer science and in many real-world applications. They are very simple to define: for any field F_q of size q, any *message length* k and *code length* n such that $k \leq n \leq q$, and any *evaluation set* of n distinct points $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\} \subseteq F_q$, the *Reed-Solomon (RS) code* $\mathbb{RS}_q(\mathcal{E}, k)$ is the set of all *codewords* $(p(\alpha_1), \ldots, p(\alpha_n))$, where $p(x) \in \mathbb{F}_q[x]$, $\deg(p) < k$.

In addition to their elegant definition and many beautiful combinatorial properties, Reed-Solomon codes also admit efficient algorithms for correcting errors. The algorithm of Berlekamp and Welch [1] corrects up to $d/2 = (n - k + 1)/2$ errors in any codeword $\mathbf{w} \in \mathbb{RS}_q(\mathcal{E}, k)$, while the *list-decoding* algorithm of Guruswami and Sudan [12] (building on groundbreaking work by Sudan [23]) can find all codewords within Hamming distance $n - \sqrt{nk}$ of a given word.

Reed-Solomon codes also play a fundamental role in modern cryptography, but are often known by a different name: *Shamir* (or *polynomial*) *secret-sharing* [21]. McEliece and Sarwate first observed [15] that sharing a secret using Shamir's scheme is equivalent to encoding the secret under an RS code: a random low-degree polynomial p is chosen so that $p(\alpha_0)$ is the value of the secret, and the shares are the evaluations of p at many other distinct points $\alpha_1, \ldots, \alpha_n$. Moreover, reconstructing the secret when players withhold or mis-report their shares is equivalent to decoding a codeword that has been corrupted with erasures or errors (respectively).

Placing shares in the exponent. Many cryptographic schemes rely on the presumed hardness of computing discrete logs in some cyclic group G of prime order q generated by an element g. In constructing threshold versions of such schemes, distributing trust over many players often involves distributing the secret key via polynomial secret sharing/RS encoding (where the alphabet is the field $\mathbb{Z}_q$). To perform the cryptographic task, typically the players must collectively compute some value of the form g^w, where w depends on the secret key and must remain secret. For example, to decrypt an ElGamal [10] ciphertext $(c, d) = (g^r, m \cdot y^r)$ where $y = g^x$ and x is the secret key, the players must collectively compute the value $c^x = g^{xr}$ without revealing their individual shares of x.

The basic protocol for computing g^w usually works as follows:

1. Player i uses its share of the secret key to compute g^{w_i}, where $w_i = p(\alpha_i)$ is a share of the secret value $w = p(\alpha_0)$ under a polynomial p of degree less than k.
2. The players broadcast their respective values of g^{w_i}, for $i = 1, \ldots, n$.
3. The broadcast values are (efficiently) "*interpolated in the exponent*"[1] to recover g^w.
   Specifically, for any $S \subset \{1, \ldots, n\}$ such that $|S| = k$, given the values $g^{w_i} = g^{p(\alpha_i)}$ for $i \in S$, each player locally computes

$$g^w = g^{p(\alpha_0)} = g^{\sum_{i \in S} \lambda_i^S p(\alpha_i)} = \prod_{i \in S} (g^{w_i})^{\lambda_i^S}$$

   using appropriate Lagrange coefficients λ_i^S:

$$\lambda_i^S = \prod_{j \in S, j \neq i} \frac{\alpha_j - \alpha_0}{\alpha_j - \alpha_i} \mod q.$$

In Step 3 above, notice that any subset S of size k suffices, and that the values from players outside S are unused in the interpolation formula. Therefore interpolation in the exponent is robust against a "halting" adversary — i.e., one that may refuse to broadcast some shares, but always correctly reports the values of those shares it does broadcast.

[1] Using the language of coding theory, we might call this "erasure-decoding in the exponent."

Introducing errors in the exponent. A *malicious* adversary, on the other hand, may lie about its shares. This introduces *errors* in the exponent, instead of erasures. Without a way to separate correct shares from incorrect shares, the interpolation formula may produce different results depending on which shares are used.

This motivates the natural question of whether it is possible to efficiently *correct errors "in the exponent."* More specifically: if a vector $\mathbf{x} = (x_1, \ldots, x_n)$ differs from some RS codeword $\mathbf{w} = (w_1, \ldots, w_n)$ in at most e positions, then given $g^{\mathbf{x}} = (g^{x_1}, \ldots, g^{x_n})$, is it possible to efficiently recover $g^{\mathbf{w}} = (g^{w_1}, \ldots, g^{w_n})$?

The goal of this paper is to investigate the computational complexity of error correction in the exponent, and to relate it to well-known computational problems in cyclic groups (such as discrete log and Diffie-Hellman).

Relationships among parameters. Error correction in the exponent involves several different parameters, and its complexity depends upon the relationships among these parameters. For analyzing asymptotic behavior, all these parameters (q, n, k, e) will be seen as functions of a single security parameter ℓ. We will focus our attention on those parameter values which are most common in cryptographic settings:

– Complexity of algorithms will always be measured relative to the security parameter ℓ. An *efficient* algorithm is one that runs in time polynomial in the security parameter. A function is said to be *negligible* if it asymptotically decreases faster than the inverse of any fixed polynomial in ℓ; otherwise it is said to be *non-negligible.*
– The alphabet size q is exponential in ℓ; that is, $q = 2^{O(\ell)}$.
– The codeword length n (which often corresponds to the number of players in a protocol) may be an arbitrary polynomial in ℓ. Therefore n is some polynomial in $\log q$.
– The message length k (which often corresponds to the number of "curious" — i.e., semi-honest — players) is at most n.
– The number of errors e (which often corresponds to the number of malicious players) is at most n.

In protocols, often it is assumed that either $e = 0$ (corresponding to an honest-but-curious adversary) or $e = k - 1$ (corresponding to a fully-malicious adversary). In order to understand the problem more generally, we will consider e and k independently.

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 efficient tools like verifiable secret sharing. In either case, these steps cost extra rounds of communication and computation, which could be avoided by instead having the parties perform local error correction (with the side-effect of also identifying cheating parties).

There are many concrete cryptographic systems in the literature which would benefit from error correction in the exponent, including (but not limited to): threshold DSS key generation and signature protocols [11], threshold ElGamal protocols [17], protocols for multiplication of shared secrets in the exponent [18], distributed pseudorandom generators, functions, and verifiable random functions [16, 9, 6], traitor-tracing schemes [2], and others. The last example of a traitor-tracing scheme is interesting because, unlike the others, it is not a threshold cryptographic protocol. This indicates that error correction in the exponent may have relevance in many other areas of cryptography as well.

On the other hand, if in our study of this problem we discover that it appears to be hard, then it can be used as a basis for new assumptions that may provide a foundation for new kinds of cryptographic schemes, or improved constructions of existing primitives.

1.2 Our Results

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

First we observe that *unique decoding* and *list decoding* in the exponent, when the number of errors e does not exceed the classical error bounds for those problems, is no harder than the computational Diffie-Hellman (CDH) problem [8] 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 $e = d - k^{1-\epsilon}$ errors is as hard as the discrete logarithm problem in the same group, for any constant $\epsilon > 0$.
- For *generic* algorithms (as defined by Shoup [22]) that only perform "black-box" group operations, 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.

Taken together, our positive and negative results may also hint at new connections between other popular problems on cyclic groups (e.g., discrete log and Diffie-Hellman), which may be illuminated by further study of error correction in the exponent.

1.3 Related Work

We are aware of only one work which directly addresses error correction in the exponent: Canetti and Goldwasser [4] gave a simple, efficient decoding algorithm which works when $e+1 = k = O(\sqrt{n})$. (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 [7].

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 [5], 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 many important ways: in their work, q is necessarily small (polynomial in n), and list-decoding is related to the discrete log problem in the field $\mathbb{F}_{q^h}$ for a somewhat large h. In contrast, we are concerned mainly with unique decoding and bounded-distance decoding as they relate to computational problems in groups of order q, where q is exponentially large in n.

Guruswami and Vardy [13] 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 many fewer errors, and for any $\mathcal{E}$.

We again stress that both of the above works [5, 13] are concerned with the hardness of plain decoding (not in the exponent).

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 $\Delta(\mathbf{x}, \mathbb{C}) = \min_{\mathbf{y} \in \mathbb{C}} \Delta(\mathbf{x}, \mathbf{y})$. Denote $\{1, \ldots, n\}$ by $[n]$.

2 Initial Observations and Upper Bounds

Unique 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. The main ingredient of the Berlekamp-Welch algorithm is simply a linear system, which can be solved in the exponent if we have a way to perform multiplication and inversion mod q (in the exponent). Multiplication is immediately provided by the Diffie-Hellman oracle: on g^a, g^b, the oracle gives us g^{ab}. Inversion can be implemented as follows: on input g^a, compute $g^{a^{-1} \bmod q} = g^{a^{q-1} \bmod q}$ by repeated squaring in the exponent. (Note that this approach requires that q be known.)

Unique decoding by enumeration. Another approach to unique decoding (under $e < (n-k+1)/2$ errors) is to merely enumerate over all subsets of size k of received shares. For each subset K, interpolate the shares (in the exponent) to each point in $\mathcal{E}$, counting the number of points in $\mathcal{E}$ for which the interpolated value disagrees with the received share. 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 in general is not polynomial in the security parameter.

A similar, but more efficient randomized approach was given in [4] 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 the same as the enumeration algorithm, except with an independent, random subset K for each iteration, for some suitable number of attempts.

Correctness of the algorithm immediately follows from the distance property of $\mathbb{RS}_q(\mathcal{E}, k)$. We now analyze the runtime: each iteration requires $O(nk \log q)$ group operations, using repeated squaring to exponentiate each share to its appropriate Lagrange coefficient. An iteration succeeds if and only if 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 options, we get a bound of:

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

which is $1/\text{poly}(n)$. Therefore the algorithm can be made to run in 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 $nk \cdot (\log q) \cdot \binom{n}{k}/\binom{n-e}{k}$.

List decoding. When the number of errors is larger than the unique decoding radius (i.e., half the distance of the code), the technique of *list decoding* can still be used to recover *all* codewords within a given radius of the received word. For example, the list decoding algorithm of Guruswami and Sudan [12] for Reed-Solomon codes can recover all codewords within a radius of $n - \sqrt{nk}$ (which is always at least as large as the unique decoding radius $(n-k+1)/2$).

The list decoding algorithm of [12] performs operations which are much more sophisticated than those of the Berlekamp-Welch unique decoding algorithm [1]. (For example, the list decoding algorithm needs to compute polynomial GCDs, perform Hensel liftings, and factor univariate polynomials.) However, it turns out that all these operations can still be performed "in the exponent" with the aid of a CDH oracle. Therefore, correcting significantly more errors (i.e., $n - \sqrt{nk}$) in the exponent also reduces to the CDH problem. (We thank abhi shelat for his assistance with these observations.)

The remainder of this paper will be devoted to establishing hardness results and 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, under a large number of errors, is as hard as the discrete log problem. We define the following code for a generator g of a cyclic group G of order q: $\mathbb{C}_q(\mathcal{E}, k, g) = \{(g^{w_1}, \ldots, g^{w_n}) : \mathbf{w} \in \mathbb{RS}_q(\mathcal{E}, k)\}$. Note that this code's alphabet is the group G. The Hamming distance Δ is defined over G^n as it is for any other alphabet.

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 $\mathbf{x}$ 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 problem of finding a non-trivial *representation* of the identity element relative to a random base, as proposed by Brands [3]:

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 FIND-REP.
Instance: A base $(x_1, \ldots, x_n) \in G^n$, chosen uniformly.
Output: Any nontrivial $(a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ such that $\prod_{i=1}^n x_i^{a_i} = 1$.

Brands showed that solving FIND-REP in G is as hard as computing discrete logs in G. For completeness, we briefly recall the result and its proof.

Proposition 3.1 ([3], Proposition 3). *If there exists an efficient randomized algorithm to solve FIND-REP in G with non-negligible probability, then there exists an efficient randomized algorithm which, on input $(g, y = g^z)$ for any generator $g \in G$ and uniform $z \in \mathbb{Z}_q$, outputs z with overwhelming probability.*

Proof. Suppose algorithm $\mathcal{B}$ solves FIND-REP in G with non-negligible probability. We construct the following 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 $\mathbb{Z}_q^n$ uniformly and independently, and let $x_i = g^{r_i} y^{s_i}$. Run $\mathcal{B}$ on $(x_1, \ldots, x_n)$, receiving correct output $(a_1, \ldots, a_n)$ with non-negligible probability. If $\sum s_i a_i \neq 0 \bmod q$, output $-\frac{\sum r_i a_i}{\sum s_i a_i} \bmod q$.

The analysis is straightforward: first observe that the constructed $(x_1, \ldots, x_n)$ is uniform over G^n, because g is a generator of prime order. Furthermore, the s_i are independent of x_i, so they are independent of $\mathcal{B}$'s output. Therefore if $(a_1, \ldots, a_n)$ is nontrivial, $\Pr[\sum s_i a_i = 0 \bmod q] = 1/q$, which is negligible. Now suppose $z = \log_g y$. Then $1 = \prod x_i^{a_i} = \prod g^{a_i(r_i + z s_i)}$, which implies $\sum a_i(r_i + z s_i) = 0 \bmod q$. Solving for z, we see that the algorithm's output is correct.

Finally, because the discrete log problem is random self-reducible, an efficient algorithm that solves discrete log with non-negligible probability can be converted into one which succeeds with overwhelming probability. $\square$

3.1 Our Reduction

Our reduction from FIND-REP to BDDE-RS relies chiefly on the following technical lemma, which bounds the probability that a *random* word in G^n (i.e., an instance of FIND-REP) is very far from an RS codeword (in the exponent). This lemma may be of independent interest, and any improvements to it will automatically reduce the error bound in our discrete log reduction.

Lemma 3.1. *For any positive integer $c \leq n - k$, and any code $\mathbb{C}_q(\mathcal{E}, k, g)$,*

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

where the probability is taken over the uniform choice of $\mathbf{x}$ from G^n.

Proof. 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 condition, which we call the "low-degree" condition for the set S:

The points $\{(\alpha_i, \log_g x_i)\}_{i \in S}$ lie on a polynomial of degree $< k$.

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 $X = \sum_{S \in \mathcal{S}} X_S$.

Now for all $S \in \mathcal{S}$, $\Pr_{\mathbf{x}}[X_S = 1] = q^{-c}$, because any k points of $\{(\alpha_i, \log_g x_i)\}_{i \in S}$ define a unique polynomial of degree at most k, and the remaining c points independently lie on that polynomial each with probability $1/q$. Then by linearity of expectation, $E[X] = \binom{n}{k+c}/q^c$. Now by Chebyshev's inequality,

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

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 and $X_{S'}$ are independent, hence they contribute little to the variance. In particular, if $|S \cap S'| \leq k$, then $E[X_S|X_{S'}=1] = E[X_S]$, i.e. X_S and $X_{S'}$ are independent and $E[X_S X_{S'}] = E[X_S]E[X_{S'}]$.

For all other distinct pairs S, S' such that $|S \cap S'| > k$, $E[X_S X_{S'}] \leq E[X_S] \leq 1/q^c$. The number of such pairs can be bounded (from above) as follows: we have $\binom{n}{k+c}$ choices for S, then $\binom{k+c}{k+1}$ choices of some $k+1$ elements of S to include in S', then $\binom{n-k-1}{c-1}$ remaining arbitrary values to complete the choice of S'. So the total number of pairs is at most $\binom{n}{k+c}\binom{k+c}{k+1}\binom{n-k-1}{c-1}$.

Putting these observations together, we obtain the following bound on σ_X^2:

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

Since $k + c \leq n$, we may apply the (very loose) bound of $\binom{n}{y} \leq n^y$ to the two binomial coefficients to get $\sigma_X^2 \leq E[X] \cdot n^{2c}$, and the claim follows. $\square$

Theorem 3.1. *For any n, k, c and q such that $\binom{n}{k+c} \geq 2q^c n^{2c}$, if an efficient randomized algorithm exists to solve $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$ with non-negligible probability (over a uniform instance and the randomness of the algorithm), then an efficient randomized algorithm exists to solve the discrete log problem in G.*

The following corollary gives concrete relationships among n, k, q, and decoding radius for which the theorem applies.

Corollary 3.1. *For any constant $\epsilon > 0$, $\delta \in (0,1]$, and any $q = 2^{O(\ell)}$ exponential in the security parameter ℓ, for any polynomial $n(\ell) = \omega(\ell^{1/\delta\epsilon})$, any $k = \Omega(n^\delta)$, $k \leq (1 - \Omega(1)) \cdot n$ and any $c \leq k^{1-\epsilon}$, the discrete log problem in cyclic groups of order q reduces to $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$.*

Example 3.1. For $k = n/2$ and $c = k^{0.99}$, we certainly have $k \leq (1 - \Omega(1)) \cdot n$ and $k = \Omega(n^1)$. Then a poly-time algorithm for bounded-distance decoding in the exponent for RS words of length $n = \ell^{100}$ under $n/2 - k^{0.99}$ errors would imply a poly-time algorithm for discrete log in groups of size about $q = 2^\ell$. In contrast, the unique decoding radius of this code is $n/4 = n/2 - k/2$, and the list decoding radius is $n - \sqrt{nk} \approx n/2 - k \cdot 0.414$; both are close to the bounded-distance radius above. Because RS codes can efficiently be uniquely- and list-decoded in the exponent using an oracle for the Diffie-Hellman problem, the error radius of our reduction comes tantalizingly close to providing a reduction from the discrete log problem to the Diffie-Hellman problem. (We thank abhi shelat for this interpretation of the result.)

Proof (of Corollary 3.1). Because $\binom{n}{k+c} \geq (\frac{n}{k+c})^{k+c}$ and $q^c \geq 2n^{2c}$ for sufficiently large ℓ, then by Theorem 3.1, it suffices to establish that for $n = \omega(\ell^{1/\delta\epsilon})$ and sufficiently large ℓ,

$$\left(\frac{n}{k+c}\right)^{k+c} \geq q^{2c} \quad \iff \quad (k+c)\log\frac{n}{k+c} \geq 2c\log q.$$

We will establish the second inequality by bounding the left side from below by $\Omega(k)$, and bounding the right side from above by $o(k)$, which suffices.

First we analyze the left term: because $c = k^{1-\epsilon}$,

$$\lim_{\ell\to\infty}\frac{n}{k+c} = \frac{n}{k} \geq 1 + \Omega(1),$$

so $\log\frac{n}{k+c} = \Omega(1)$. Therefore the left term is $\Omega(k)$.

On the right, we have $2c\log q = c \cdot O(\ell)$. Because $c \leq k^{1-\epsilon}$ and $n = \omega(\ell^{1/\delta\epsilon}) \iff \ell = o(n^{\delta\epsilon})$, the right side is $k^{1-\epsilon}\cdot o(n^{\delta\epsilon})$. Finally $n^\delta = O(k)$, so we get $k^{1-\epsilon}\cdot o(k^\epsilon) = o(k)$, as desired. $\square$

Proof (of Theorem 3.1). Suppose that algorithm $\mathcal{D}$ solves $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$ with non-negligible probability. By Proposition 3.1, it will suffice to construct an algorithm $\mathcal{A}$ that solves $\mathsf{FIND\text{-}REP}$ in G with non-negligible probability.

$\mathcal{A}$ works as follows: on input $\mathbf{x} = (x_1, \ldots, x_n)$, where $\mathbf{x}$ is uniform over G^n, immediately run $\mathcal{D}(g, \mathbf{x})$. By Lemma 3.1, $(g, \mathbf{x})$ is an instance of $\mathsf{BDDE\text{-}RS}_{q,\mathcal{E},k,n-k-c}$ with probability at least $1/2$. Then conditioned on this event, the instance is uniform, and with non-negligible probability $\mathcal{D}$ outputs 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 $i \in E$. Then any k of the $\mathbf{x}_i$ linearly interpolate (in the exponent) to the remaining x_i. That is, we can compute non-trivial Lagrange coefficients 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 $(\lambda_1, \ldots, \lambda_n)$, which is a solution to $\mathsf{FIND\text{-}REP}$. $\square$

4 Generic Algorithms for Noisy Polynomial Interpolation

Generic algorithms. Shoup proposed the *generic algorithms* framework [22] for computational problems in groups. 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.

Schwartz's lemma. A key tool in the analysis of generic algorithms is *Schwartz's Lemma*, which bounds the probability that a multivariate nonzero polynomial, defined over a finite field, is zero at a random point.

Lemma 4.1 ([20]). *For any nonzero polynomial $f \in \mathbb{F}_q[X_1, \ldots, X_t]$ of total degree d,*

$$\Pr[f(x_1, \ldots, x_t) = 0] \leq d/q,$$

where the probability is taken over a uniform choice of $(x_1, \ldots, x_t) \in \mathbb{F}_q^t$.

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 4.1 is potentially stronger than one which might be provided for the full-decoding task.

Theorem 4.1. *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 4.1. *If $ek = \omega(n \log n)$, then no efficient generic algorithm solves $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$, except with probability negligible in the security parameter. In particular, the algorithm of Canetti and Goldwasser [4] (described in Section 2) is optimal.*

Proof (of Corollary 4.1). 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, and hence in the security parameter. Since $1/q$ is negligible as well, the total success probability is negligible. □

Proof (of Theorem 4.1). 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 Lemma 4.1, 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. $\square$

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 [14], 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 4.2. *A generic algorithm for $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$, augmented with a DDH oracle, 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 4.2. *If $ek = \omega(n \log n)$, no efficient generic algorithm augmented with a DDH oracle solves $\mathsf{GNPI}_{q,\mathcal{E},\alpha_0,k,e}$, except with probability negligible in the security parameter.*

Proof (Sketch of Theorem 4.2). As in the proof of Theorem 4.1, 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 $\mathbb{Z}_q[P_0, \ldots, P_{k-1}, E_1, \ldots, E_n]$. (Because every F_i is linear, the ideal game will only respond "yes" when at least one of F_a, F_b is a constant.)

The failure event $\mathcal{F}$ is the union of the old failure event (from the proof of Theorem 4.1) 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 Lemma 4.1, $\Pr[F'(\mathbf{p}, \mathbf{e}') = 0] \leq 2/q$.

It remains to bound $\Pr_{\mathbf{e}}[F' = 0 \mid F \neq 0]$. In order to have $F \neq 0$ and $F' = 0$, we consider two mutually exclusive cases: (1) F_a or F_b (or both) is a constant polynomial, or (2) F_a, F_b are both non-constant polynomials, i.e. of positive degree.

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 4.1, $\Pr[F' = 0 \mid F \neq 0] \leq \binom{n-k}{e} / \binom{n}{e}$.

For case (2), we first introduce some notation: for a polynomial H and a monomial Z, define $\text{coeff}_Z(H)$ to be the coefficient of Z in H. We claim that for either $i = a$ or $i = b$, F'_i is a constant polynomial. Suppose not: then there exist two indeterminants X, Y such that $\text{coeff}_X(F'_a) \neq 0$ and $\text{coeff}_Y(F'_b) \neq 0$. If $X = Y$, we see that $\text{coeff}_{X^2}(F') \neq 0$, a contradiction. If $X \neq Y$, we have

$$\text{coeff}_{XY}(F') = \text{coeff}_X(F'_a)\text{coeff}_Y(F'_b) + \text{coeff}_X(F'_b)\text{coeff}_Y(F'_a) = 0.$$

Then $\text{coeff}_X(F_b) \neq 0$, which implies that $\text{coeff}_{X^2}(F') \neq 0$, a contradiction.

Using reasoning as in the proof of Theorem 4.1, we see that

$$\Pr[F'_a \text{ or } F'_b \text{ is constant} \mid F_a, F_b \text{ are non-constant}] \leq 2\binom{n-k}{e} / \binom{n}{e}.$$

Taking a union bound over all queries to the DDH oracle, we get the claimed result. $\square$

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.

Problem: Decisional 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 $\mathsf{DGNPI}_{q,\mathcal{E},\alpha_0,k,e}$.
Instance: Encoding list $(\sigma(P(\alpha_1)+e_1), \ldots, \sigma(P(\alpha_n)+e_n), \sigma(1), \sigma(z_0), \sigma(z_1))$ for a random $P(x) \in \mathbb{Z}_q[x]$, $\deg(P) < k$, a random $\mathbf{e} \in \mathbb{Z}_q^n$ such that $\mathrm{wt}(\mathbf{e}) = e$, and a random bit b where $z_b = P(\alpha_0)$ and z_{1-b} is random.
Output: The bit b.

Theorem 4.3. *A generic algorithm for $\mathsf{DGNPI}_{q,\mathcal{E},\alpha_0,k,e}$ making m queries succeeds with probability at most $\frac{1}{2} + 2m^2 \left(1/q + \binom{n-k}{e}/\binom{n}{e}\right)$.*

Proof (Sketch). The proof is very similar to the proof of Theorem 4.1. We again imagine a game which maintains a list of polynomials F_i in the indeterminants $P_0, \ldots, P_{k-1}, E_1, \ldots, E_n$, and two new indeterminants Z_0, Z_1. In the ideal game, the two input polynomials corresponding to z_0 and z_1 are just Z_0 and Z_1, respectively. In the ideal game, every distinct polynomial is mapped to a different string, and the algorithm succeeds with probability 1/2 because its view is independent of b. The failure event is that for some $F_i \neq F_{i'}$, either $F(\mathbf{p}, \mathbf{e}, \sum_{j=0}^{k-1} p_j \alpha_0^j, z) = 0$ or $F(\mathbf{p}, \mathbf{e}, z, \sum_{j=0}^{k-1} p_j \alpha_0^j) = 0$ where $F = F_i - F_{i'}$ and z is chosen at random. From here, the analysis proceeds as in Theorem 4.1. □

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 full 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:

– Find some other family of codes which admits an efficient (preferably generic) algorithm for decoding in the exponent, and which can be used as the basis of a secret-sharing scheme — or show that the two goals are mutually incompatible.

– Demonstrate a non-generic decoding algorithm for a specific class of cyclic groups with performance better than the generic bounds (perhaps using ideas from index calculus methods).
– Provide new constructions of standard (or new) cryptographic primitives, assuming error correction in the exponent is hard. Such constructions would be useful both as a hedge against possible attacks on other (stronger) assumptions, and for any unique functionality properties they may have.
– Show new connections between the discrete log and Diffie-Hellman problems, using the fact that decoding is often easy with a CDH oracle.

In addition, the general idea of correcting errors in "partially hidden" data (i.e., data that has been obscured by some one-way function) seems ripe with interesting problems.

Acknowledgements

The author gratefully thanks Shafi Goldwasser, Ran Canetti, Alon Rosen, Adam Smith, Tal Rabin, and abhi shelat for helpful comments and discussions, and the anonymous reviewers for their valuable and constructive suggestions.

References

1. E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.
2. D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 338–353, London, UK, 1999. Springer-Verlag.
3. S. Brands. Untraceable off-line cash in wallet with observers. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 302–318, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
4. 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.
5. 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.
6. R. Cramer and I. Damgård. Secret-key zero-knowlegde and non-interactive verifiable exponentiation. In *1st TCC*, pages 223–237, 2004.
7. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO'98*, 1998.
8. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
9. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *6th PKC*, pages 1–17, 2003.
10. T. E. Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

11. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold dss signatures. In *Advances in Cryptology — Eurocrypt '96*, pages 354–371, 1996.
12. 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.
13. V. Guruswami and A. Vardy. Maximum-likelihood decoding of Reed-Solomon codes is NP-hard. In *SODA*, 2005.
14. A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.
15. R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM*, 24(9):583–584, 1981.
16. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and kdcs. In *Advances in Cryptology — Eurocrypt '99*, pages 327–346, 1999.
17. C. Park and K. Kurosawa. New ElGamal type threshold digital signature scheme. *IEICE Trans. Fundamentals*, E79-A(1):86–93, January 1996.
18. M. D. Raimondo and R. Gennaro. Secure multiplication of shared secrets in the exponent. Cryptology ePrint Archive, Report 2003/057, 2003.
19. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8(2):300–304, June 1960.
20. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
21. A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, 1979.
22. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. Eurocrypt '97*, pages 256–266, 1997.
23. M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.