Provable Security Properties in Generic Groups

David Philipson

1 Introduction

One of the main motivations for moving to elliptic curve groups in cryptography is to avoid the computational properties of simpler groups such as $\mathbb{F}_p^\times$, which are open to a range of attacks due to their well-understood relation to the integers. Take, for instance, the discrete log problem on a group of p elements, whose difficulty is the basis for a number of cryptographic applications, such as ElGamal. While we can solve this problem in $\mathbb{F}_p^\times$ in subexponential time using the index calculus, if we attempt to solve the smae problem without relying on specific computational properties of $\mathbb{F}_p^\times$, we must instead use an algorithm such as Pollard-rho, which has an exponential running time of $\Omega(\sqrt{p})$. In this report, I review the work of Shoup and the work of Boneh and Boyen, which examines the limitations of such "generic algorithms." In particular, Shoup demonstrates that we cannot hope to perform asymptotically better than our known algorithms for solving the discrete log or Diffie-Hellman problems using generic algorithms. Boneh and Boyen then allow the algorithms to make use of pairings, and demonstrate that they still have limitations which are useful for cryptography.

2 Definitions and preliminaries

The definitions in this section are taken from Shoup's paper, "Lower Bounds for Discrete Logarithms and Related Problems."

We can consider a generic algorithm to be an algorithm which answers questions about a cyclic group $\mathbb{Z}/n$, but rather than interacting directly with the group, it only interacts with encodings of the group elements through some unknown encoding function. At any time, the algorithm may make queries to a "group operation oracle" which will perform a group operation and respond with the encoding of the result. Hence, the algorithm is unable to make use of any computations in the group other than those based on the group law. Formally, generic algorithms are defined as follows:

Definition. A **generic algorithm** $\mathcal{A}$ for $\mathbb{Z}_n$ on S is a probabilistic algorithm that behaves as follows: it takes as input an **encoding list** $(\sigma(x_1), \ldots, \sigma(x_k))$, where each x_i is in $\mathbb{Z}_n$, and σ is an encoding function of $\mathbb{Z}/n$ on S. As the algorithm executes, it may from time to time consult an **oracle**, specifying two indices i and j into the encoding list, and a sign bit. The oracle computes $\sigma(x_i \pm x_j)$, according to the specified sign bit, and this bit string is appended to the encoding list (to which $\mathcal{A}$ always has access). The output of $\mathcal{A}$ is a bit string denoted $\mathcal{A}(\sigma; x_1, \ldots, x_k)$.

We will be interested in the performance of generic algorithms on the discrete log problem and both the computational and decision Diffie-Hellman problems, which we now define.

Definition. Let G be a cyclic group with n elements, which we write multiplicatively (as we will all groups throughout this paper other than the groups $\mathbb{Z}/n$, which we write additively). Let g be a generator of G and let $k \in [0, n)$. Given (g, g^x), the **discrete log problem** (**DLP**) is to compute x.

Definition. Let G be a cyclic group with n elements. Let g be a generator of G and let $x, y \in [0, n)$. Given (g, g^x, g^y), the **computational Diffie-Hellman problem** (**CDH**) is to compute g^{xy}.

Definition. Let G, n, g, x, y be as in the previous definition, and let $r \xleftarrow{R} [0, n)$ and then $z \xleftarrow{R} \{xy, r\}$. Given (g, g^x, g^y, g^z), the **decision Diffie-Hellman problem** (**DDH**) is to determine whether $z = xy$ or r.

3 Fully generic algorithms

In his paper "Lower Bounds for Discrete Logarithms and Related Problems," Shoup demonstrates a series of results relating to the limitations of generic algorithms on the problems given above. His major result on the discrete log problem is the following.

Theorem. *Let n be a positive integer whose largest prime divisor is p. Let $S \subseteq 0, 1^*$ be a set of cardinality at least n. Let A be a generic algorithm for $\mathbb{Z}/n$ on S that makes at most m oracle queries. If $x \in \mathbb{Z}/n$ and an encoding function σ are chosen at random, then the probability that $\mathcal{A}(\sigma; 1, x) = x$ is $O(m^2/p)$.*

Note that $A(\sigma; 1, x) = x$ is equivalent to stating that the algorithm has solved the discrete log problem, since the algorithm was given the encodings of the elements 1 and x and was able to determine that the second element is the first one combined with itself x times. Further, suppose we desire some constant lower bound c on our algorithm's accuracy $O(m^2/p)$. Solving for the the number of required oracle queries m tells us that we need at least $\Omega(\sqrt{p})$ queries, matching the number used by Pollard-rho, for instance. Finally, note that the fact that this theorem holds over random encodings σ implies that there is in particular at least one encoding for which the probability of success is at most $O(m^2/p)$. Since we only need one "bad" encoding to demonstrate the weakness of the generic algorithm, this theorem implies the result discussed above: that generic algorithms cannot solve the discrete log problem in faster than $\Omega(\sqrt{p})$ time.

To prove the theorem, we first need a lemma.

Lemma. *Let p be prime and let $t \geq 1$. Let $F(X_1, \ldots, X_k) \in (\mathbb{Z}/p^t)[X_1, \ldots, X_k]$ be a nonzero polynomial of degree d. Then for random $x_1, \ldots, x_k \in Z/p^t$, the probability that $F(x_1, \ldots, x_k) = 0$ is at most d/p.*

Proof of lemma. For $t = 1$, assume without loss of generality that there is only one variable x (if there are more, first choose random $x_2, \ldots, x_k$, producing a polynomial in one variable). Since this is a degree at most d polynomial in $\mathbb{F}_p$, it has at most d distinct roots, hence there there is at most a d/p chance that a randomly selected x produces $F(x) = 0$. For $t > 1$, first divide out the equation $F = 0$ by the highest possible power of P, and thus the image of F in $\mathbb{Z}/p$ is nonzero. Since $x_1, \ldots, x_k$ are selected from $\mathbb{Z}/p^t$ according to a uniform distribution, so too are their images in $\mathbb{Z}/p$, and so we can use the result from when $t = 1$. □

Proof of theorem. Write $n = p^t s$ where p is prime and $p \nmid s$. Rather than providing a real oracle for the algorithm, we allow it to interact with a simulated oracle which effectively makes up an encoding as it goes along and waits until the end before choosing x. The simulated oracle follows the following rules:

- The simulated oracle stores a list $F_1, \ldots, F_k$ of polynomials in $\mathbb{Z}/p^t[X]$, a list $z_1, \ldots, z_k$ of values in $\mathbb{Z}/s$, and a list $\sigma_1, \ldots, \sigma_k$ of values in S. This final list is the values it has given to the adversary.
- To initialize, we take $k = 2$. The first list has elements $1, X$. The second list has elements $1, z$ where $z \xleftarrow{R} \mathbb{Z}/s$. The third list has σ_1, σ_2, where both $\sigma_1, \sigma_2 \xleftarrow{R} S$, subject to $\sigma_1 \neq \sigma_2$.
- When the adversary makes a query with indices i and j, we append values to the three lists as follows:

  - $F_{k+1} = F_i \pm F_j$.
  - $z_{k+1} = z_i \pm z_j$.
  - If the pair F_{k+1}, z_{k+1} has already appeared at an earlier index l (that is, there exists $l \leq k$ with $F_{k+1} = F_l$ and $z_{k+1} = z_l$, then $\sigma_{k+1} = \sigma_l$. Otherwise, select $\sigma_{k+1} \overset{R}{\leftarrow} S$.

- Suppose the adversary algorithm outputs $y \in Z/n$ upon termination, and let y' be the image of y in $\mathbb{Z}/p^t$. Select a random $x \in \mathbb{Z}/p^t$. We say that the adversary has won if $x = y'$, or if $F_i(x) = F_j(x)$ for any $F_i \neq F_j$.

We first find an upper bound on the probability that the adversary wins. Note that for any fixed i, j with $F_i \neq F_j$, if we set $F = F_i - F_j$ then we have that $F \neq 0$. Since the degree of F is at most 1 (since all F_i are linear combinations of 1 and X), we have by the lemma that the probability that $F(x) = 0$ is at most $1/p$. Since there are $O(m^2)$ such pairs i, j and further the probability that $x = y'$ is at most $1/p$, we have that the probability of the adversary winning is indeed $O(m^2/p)$.

Now, observe that as long as the adversary does *not* win the game, then the output of the simulated oracle is indistinguishable from that of a real oracle with random x and encoding function, in a scenario in which the adversary does not output a correct answer. This occurs by construction, since the first case in which the adversary wins correponds to a correct response to the discrete log problem, while the remaining cases correspond to situations where the simulated oracle has incorrectly presented two equal group elements as nonequal. On the other hand, as long as the adversary did not win, the lists used by the simulated oracle ensure that all outputs are consistent with those under some selection of x and some encoding, both of which are uniform over their respective spaces. Hence, the probability that the algorithm outputs the correct answer is no greater than the probability that the algorithm wins the game, completing the proof. $\square$

Shoup also produces a similar bound for both the computational and decision Diffie-Hellman problems. The results and their proofs are as follows.

Theorem. *Let n be a positive integer whose largest prime divisor is p. Let $S \subseteq \{0,1\}^*$ be a set of cardinality at least n. Let A be a generic algorithm for $\mathbb{Z}/n$ on S that makes at most m oracle queries. If $x, y \in \mathbb{Z}/n$ and an encoding function σ are chosen at random, then the probability that $A(\sigma; 1, x, y) = \sigma(xy)$ is $O(m^2/p)$.*

Proof. The proof is very similar to that of the previous theorem. Begin with the same set-up as the previous proof, and make the following changes to the simulated oracle:

- The polynomials in the first list now are in two variables, X and Y.

- The lists begin with three elements each. The first list begins with elements $1, X, Y$. The second list begins with $1, z_x, z_y$, where $z_x, z_y \overset{R}{\leftarrow} \mathbb{Z}/s$. The third list begins with $\sigma_1, \sigma_2, \sigma_3$, where $\sigma_1, \sigma_2, \sigma_3 \overset{R}{\leftarrow} S$.

- When the algorithm terminates, we pick $x, y \overset{R}{\leftarrow} \mathbb{Z}/p^t$. We say that the adversary has won if $F_i(x, y) = F_j(x, y)$ for some $F_i \neq F_j$ or if $F_i(x, y) = xy$ for some i.

As before, by the lemma the probability that $F_i(x, y) = F_j(x, y)$ for fixed i, j is at most $1/p$. Also by the lemma, the probability that $F_i - XY$ vanishes at (x, y) is at most $2/p$. Hence, the probability that the adversary wins is again $O(m^2/p)$.

We may assume that the adversary's output is one of the "encodings" from the oracle, since otherwise its response is no better than a random guess, and hence has probability of success bounded by $1/(p-m)$. Supposing that the adversary does indeed output one of the encodings it has received, we have as in the previous proof that in all cases where the adversary does *not* win the game, the simulated oracle's output is indistinguishable from that of a true oracle, in a case where the algorithm's output was incorrect. Again, we have that the probability of a correct answer is therefore no greater than the probability of winning the game, hence $O(m^2/p)$. □

Theorem. *Let n be a positive integer whose smallest prime divisor is p. Let $S \subseteq \{0,1\}^*$ be a set of cardinality at least n. Let A be a generic algorithm for $\mathbb{Z}/n$ on S that makes at most m oracle queries. Let $x, y, z \in Z/n$ be chosen at random, let σ be a random encoding function, and let b be a random bit. Also, let $w_0 = xy$ and $w_1 = z$. Then the probability that $A(\sigma; 1, x, y, w_b, w_{1-b}) = b$ is $1/2 + O(m^2/p)$.*

Proof sketch. The proof is once again quite similar, so we give only an outline. We modify the simulated oracle once again. This time, our polynomials will be in four variables, X, Y, U, V. Upon the adversary's final response, we select random x, y, z, and say that the adversary wins if $F_i(x, y, z, xy) = F_j(x, y, z, xy)$ or $F_i(x, y, xy, z) = F_j(x, y, xy, z)$ for some $F_i \neq F_j$. We can show by a similar argument to the above that the probability the adversary wins is $O(m^2/p)$, and the probability that the algorithm determines b is bounded by $1/2$ plus the probability that the algorithm wins this game. □

It is worth noting that in the case of some elliptic curve groups (with small embedding degree), we can actually break DDH using the non-generic property of easily computable Weil or Tate pairings. In particular, given $(P, [a]P, [b]P, [c]P)$ where c is either ab or random and $P \in E[n]$, we can compute pairings $e([a]P, [b]P) = e(P, P)^{ab}$ and $e(P, [c]P) = e(P, P)^c$. If these two results are the same, then we may output that $c = ab$. The fact that elliptic curve groups admit pairing-based attacks suggests that we might find it more suitable to examine the capabilities of generic groups which also have computable pairings. This is the subject of Boneh and Boyen's paper, and the following sections.

4 Generic groups with pairings and the Strong Diffie-Hellman assumption

In their paper "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups," authors Boneh and Boyen present several signature schemes which are based on groups with pairings and rely on a specific security assumption, which they name the Strong Diffie-Hellman assumption (SDH). These signature methods rely on groups in which it is both easy to solve the DDH problem, as discussed at the end of the previous section, and yet hard to solve the Strong Diffie-Hellman problem. In order to argue that their schemes are secure, Boneh and Boyen prove that using an expanded definition of generic algorithms, which are allowed to make use of a pairing as well as the group operation, the SDH assumption holds. We will continue our examination of generic algorithms by tracing this result.

First, we need some definitions, which are taken from Boneh and Boyen's "Short Signatures without Random Oracles and the SDH Assumption."

Definition. If $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, then $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a **bilinear pairing** if it satisfies:

- **bilinearity** – for all $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ and for all $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$, and
- **non-degeneracy** – if g_1, g_2 are respectively generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, then $e(g_1, g_2) \neq 1$ and is thus a generator of $\mathbb{G}_T$.

Definition. We say that $(\mathbb{G}_1, \mathbb{G}_2)$ are a **bilinear group pair** if there exists a group $\mathbb{G}_T$ and a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, and the pairing e and the group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are all efficiently computable.

These defintions allow the possibility of distinct $\mathbb{G}_1, \mathbb{G}_2$ in order to be general, although the simpler case $\mathbb{G}_1 = \mathbb{G}_2$ is entirely possible, as in the case of the Weil pairing on supersingular curves.

Definition. Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic groups of prime order p, generated respectively by g_1, g_2. In the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, the **$q$-Strong Diffie-Hellman** problem (**q-SDH**) is as follows: given as input a $(q+3)$-tuple of elements

$$(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2,$$

produce as output a pair $(c, g_1^{1/(x+c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$ for a freely chosen value $c \in \mathbb{Z}_p \setminus \{-x\}$.

Note that in the case $\mathbb{G}_1 = \mathbb{G}_2$, the last two elements of the input to the SDH problem are redundant and can be omitted.

The SDH problem is useful for signature schemes due to the property that it has random self-reduction. That is, given one instance of the SDH problem, we may produce a random additional SDH problem instance where the solution to the second allows us to solve the first. Boneh and Boyen use this property as the basis for several signature schemes, but such applications are not the focus of this paper.

We must also update our definition of a generic algorithm to allow it to operate on bilinear group pairs. As before, we may conceptualize such an algorithm as one which interacts with the underlying groups only through the outputs an arbitrary and unknown encoding, while making queries to an oracle to perform the underlying group and pairing operations. We adjust the previous definition to work on a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with respective generators g_1, g_2 and pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as follows.

- We now have three encodings $\sigma_1, \sigma_2, \sigma_3\colon \mathbb{Z}/p \to S$ for the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively.
- Likewise, we now have three oracles for the group laws in each of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$.
- We have an oracle for computing the pairing. That is, if the algorithm has received the encodings $\sigma_1(x)$ and $\sigma_2(y)$ for $x, y \in \mathbb{Z}/p$, it may request from the oracle the encoding $\sigma_T(xy)$.
- We also grant the algorithm oracles for computing a homomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. Supposing without loss of generality that this homomorphism sends g_2 to g_1, we may say that if the algorithm has received the encoding $\sigma_2(x)$, then it may request from the oracle $\sigma_1(x)$. Likewise, we give the algorithm for the reverse homomorphism ψ^{-1}. Note that in practice, such homomorphisms may not be computable in certain groups, but since granting the adversary these oracles only makes it stronger, we may do so for our proof.

5 Generic security of the SDH assumption

The main result of Boneh and Boyen is as follows.

Theorem. *Suppose $\mathcal{A}$ is an algorithm that solves the q-SDH problem in generic bilinear groups of order p, making at most q_G oracle queries for the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, the homomorphisms $x \in \mathbb{Z}_p^\times$, and the bilinear pairing e, all counted together. Suppose also that the integer $x \in \mathbb{Z}_p^\times$*

and the encoding functions $\sigma 1, \sigma_2, \sigma_T$ are chosen random. Then, the probability ϵ, that $\mathcal{A}$ on input $(p, \sigma_1(1), \sigma_1(x), \ldots \sigma_1(x^q), \sigma_2(1), \sigma_2(x))$ outputs $(c, \sigma_1(\frac{1}{x+c}))$ with $c \in \mathbb{Z}_p \setminus \{-x\}$, is bounded as

$$\epsilon \leq \frac{(q_G + q + e)^2(q+1)}{p} \leq O\left(\frac{q_G^2 q + q^3}{p}\right).$$

This theorem implies that if we wish to ensure some constant lower bound on our probability of success in generic bilinear groups with $q < O(\sqrt[3]{p})$, then our generic algorithm must make at least $\Omega(\sqrt{p/q})$ queries.

Proof. Our proof will once again take a similar structure, where we allow the adversary to interact with a set of simulated oracles which make up encodings as they go along and choose the value of x only at the very end. The simulated oracles work as follows:

- The oracles maintain three lists of pairs, $L_1 = \{(F_{1,i}, \sigma_{1,i})\}_{i=1,\ldots,\tau_1}$, $L_2 = \{(F_{2,i}, \sigma_{2,i})\}_{i=1,\ldots,\tau_2}$, and $L_3 = \{(F_{T,i}, \sigma_{T,i})\}_{i=1,\ldots,\tau_T}$. The $F_{1,i}$ and $F_{2,i}$ are polynomials in $(\mathbb{Z}/p)[X]$ of degree at most q, while $F_{T,i}$ are polynomials in $(\mathbb{Z}/p)[X]$ of degree at most $2q$. The values $\sigma_{1,i}, \sigma_{2,i}, \sigma_{T,i}$ are the "encodings" which are given to the adversary.

- To initialize, we take $\tau_1 = q+1$, $\tau_2 = 2$, and $\tau_T = 0$. We initialize L_1 with the pairs $(X^{i-1}, \sigma_{1,i})$ for $i = 1, \ldots, q$, where each $\sigma_{1,i}$ is selected uniformly at random from S. Likewise, we initialize L_2 with the pairs $(X^{i-1}, \sigma_{2,i})$ for $i = 1, 2$.

- The oracles respond to queries according to the following rules:

  group operations The same as in the earlier algorithms. If the adversary requests that the group operation be performed for indices i, j in $\mathbb{G}_1$, we set $F_{i,\tau_1+1} = F_{1,i} + F_{1,j}$. If this is a polynomial which has already occured at index k (i.e. $F_{1,\tau_1+1} = F_{1,k}$ with $k \leq \tau_1$), then we set $\sigma_{1,\tau_1+1} \leftarrow \sigma_{1,k}$, otherwise we select $\sigma_{1,\tau_1+1} \overset{R}{\leftarrow} S$. We then pass σ_{1,τ_1+1} back to the adversary and increment τ_1. The rules for group operations are equivalent in $\mathbb{G}_2$ and $\mathbb{G}_T$.

  pairing Suppose the adversary queries result of the pairing with $\sigma_{1,i}$ and $\sigma_{2,j}$. The simulated oracle sets $F_{T,\tau_T+1} = F_{1,i} \cdot F_{2,j}$. If this polynomial already appeared in L_T at index k, i.e. $F_{T,\tau_T+1} = F_{T,k}$ with $k \leq \tau_T$, then we set $\sigma_{T,\tau_T+1} \leftarrow \sigma_{T,k}$. Otherwise, we choose a random $\sigma_{T,\tau_T+1} \overset{R}{\leftarrow} S$. We pass σ_{T,τ_T+1} back to the adversary and increment τ_T.

  homomorphisms Take the case of a homomorphism query from $\mathbb{G}_2$ to $\mathbb{G}_1$, where the adversary queries with string $\sigma_{2,i}$. The simulated oracle sets $F_{1,\tau_1+1} = F_{2,i}$. If L_1 already contained a copy of this polynomial, i.e. $F_{1,\tau_1+1} = F_{1,j}$ for some $j \leq \tau_1$, then we set $\sigma_{1,\tau_1+1} \leftarrow \sigma_{1,k}$. Otherwise, we choose a random $\sigma_{1,\tau_1+1} \overset{R}{\leftarrow} S$. We pass σ_{1,τ_1+1} back to the adversary and increment τ_1.

- Once $\mathcal{A}$ terminates and returns a pair $(c, \sigma_{1,\ell}$, we let $F_{1,\ell}$ be the corresponding polynomial in L_1. Let $F_{T,\star} = F_{1,\ell} \cdot (X + c)$. We select a random $x \overset{R}{\leftarrow} \mathbb{Z}/p$, then announce that the adversary has "won" if $F_{T,\star}(x) = 1$, or if there are any two nonequal polynomials in L_1 which take the same value at x, or the same condition in L_2 or in L_3.

Examining the win condition, note that $F_{T,\star} = F_{1,\ell} \cdot (X + c) = F_{1,\ell} \cdot (F_{2,2} + c \cdot F_{2,j})$. Hence, the equality $F_{T,\star}(x) = 1$ corresponds to the relation $e(A, g_2^{x+c}) = e(g_1, g_2)$ where A is the element of $\mathbb{G}_1$

represented by $\sigma_{1,\ell}$; in this case, the adversary has successfully answered the SDH problem. Hence, as long has the adversary does *not* win, the simulated oracles' outputs are indistinguishable from those of real oracles in a scenario where the adversary was not successful. Therefore, the probability that the the $\mathcal{A}$ sucessfully solves SDH is bounded above by the probability that it wins the game.

We compute the probability that $\mathcal{A}$ wins the game. Since the degree of $F_{T,\star}$ is at least 1 and at most $q+1$, we have by the lemma that the probability that $F_{T,\star}(x) = 1$ for uniformly random x is at most $(q+1)/p$. Further, the polynomials in L_1, L_2, L_T are all of degree at most $2q$, and so the probability that any fixed pair in one of the lists is equal at a uniformly random x is at most $2q/p$, again by the lemma. Thus, the probability that $\mathcal{A}$ wins is at most

$$\left(\binom{\tau_1}{2} + \binom{\tau_2}{2} + \binom{\tau_T}{2}\right)\frac{2q}{p} + \frac{q+1}{p}.$$

Since each query increments exactly one of τ_1, τ_2, τ_T and initially we have $\tau_1 + \tau_2 + \tau_T = q+3$, we have that $\tau_1 + \tau_2 + \tau_T \leq q_G + q + 3$, where q_G is the number of queries. Hence, the probability ϵ that $\mathcal{A}$ is successful is bounded by

$$\epsilon \leq \frac{(q_G + q + 3)^2(q+1)}{p} \leq O\left(\frac{q_G^2 q + q^3}{p}\right),$$

which is the desired result. $\square$

6 Conclusion

We have demonstrated a number of limitations on generic algorithms, even when they are allowed to compute pairings. These results are helpful on two fronts. First, they inform us that in order to surpass current known running times on thse problems, we must find a way to take advantage of specific group properties. Second, they help convince us that our security assumptions are somewhat reasonable, which allows us to more confidently use them as the basis for cryptographic schemes. Although they cannot guarantee security since we do not have real-word generic groups, they help guide our understanding of algorithmic limitations.

7 References

V. Shoup. Lower bounds for discrete logarithms and related problems. *EUROCRYPT 1997*.

D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *EUROCRYPT 2004*.