

Cryptography and the Variational Stability of Algorithms

JOSEPH H. SILVERMAN

ABSTRACT. Many algorithms exhibit a wide variation of running times when presented with different inputs. For such algorithms, if the goal is simply to solve a single problem instance, then it may be more efficient to set a cutoff time and to start on a new problem instance if the chosen cutoff time is exceeded. Whether or not this cutoff strategy is helpful depends on the extent to which the running time varies. In this note we quantify this notion of algorithmic variability and we define a stability exponent StExp with the property that a cutoff strategy is useful if and only if $\text{StExp} > 1$. We compute the stability exponent exactly for exhaustive searches and for meet-in-the-middle (e.g., Pollard rho) searches and we estimate the stability exponent experimentally for an LLL lattice reduction implementation. These three examples have applications, respectively, to symmetric ciphers (DES, AES), elliptic curve cryptosystems (ECC), and lattice cryptosystems (NTRU).

INTRODUCTION

We begin by giving an unrigorous description of the problem that we will study in this note. In order to make our problem mathematically precise, we will need to index our algorithms by the size of their inputs, but for now we ignore this complication.

Let \mathcal{A} be an algorithm that solves a certain class of problems \mathcal{P} and let T be the random variable that describes the running time of the algorithm. That is, for any particular problem $\omega \in \mathcal{P}$, the quantity $T(\omega)$ is the time it takes the algorithm \mathcal{A} to solve ω . The running time may vary significantly for different problems in \mathcal{P} , and it may happen that we will be satisfied with solving a single instance of a problem in \mathcal{P} . More precisely, we assume that for any K , there is a person who is willing to provide us with K different problems from \mathcal{P} , and we consider our efforts successful if we are able to solve any one of the K problems.

Date: November 2003, Version 1.

1991 Mathematics Subject Classification. Primary: 68Q25; Secondary: 68W40.

Key words and phrases. running time of algorithms, cryptography, AES, ECC, NTRU.

The author would like to thank Edlyn Teske for her assistance.

If the random variable T is constant, we can do no better than to take a single problem and work on it until it is solved. On the other hand, if the running times for different problems in \mathcal{P} fluctuate significantly, then the following cutoff strategy might yield an improvement:

- (1) Choose a cutoff time C .
- (2) Request a problem $\omega \in \mathcal{P}$.
- (3) Run the algorithm \mathcal{A} on ω either until the problem is solved or until the time expended equals C .
- (4) If the problem ω was not solved in Step 3, go back to Step 2 and request a new problem.

Suppose that for the chosen C , the cutoff algorithm requires K iterations of Step 2 (on average) before solving a problem. Thus if we let $T_1, T_2, T_3, \dots, T_K$ be independent random variables with the same distribution as T , then we are saying that on average,

$$\min\{T_1, T_2, \dots, T_K\} \approx C.$$

More precisely, if we let

$$M_K = \min\{T_1, T_2, \dots, T_K\},$$

then the cutoff algorithm will succeed in K steps (on average) if

$$E(M_K) \approx C.$$

If this happens, then the total running time will be approximately KC . (More precisely, there will be $K - 1$ iterations of Step 3, each of which takes time C , plus the final iteration that takes at most time C .) The conclusion is that

$$\text{Running Time of Cutoff Algorithm} \approx KE(M_K).$$

For many algorithms, one finds (at least asymptotically) that

$$E(M_K) \approx K^{-\epsilon(\mathcal{A})}$$

for some exponent $\epsilon(\mathcal{A})$ that is more-or-less constant as K varies. We call this exponent the *Stability Exponent* of the algorithm \mathcal{A} . Then the running time of the cutoff algorithm is approximately $K^{1-\epsilon(\mathcal{A})}$, so the cutoff algorithm is helpful if the stability exponent is greater than 1, and it is not helpful if the stability exponent is smaller than 1.

Our aim in this paper is to give a more precise, mathematically rigorous definition of the stability exponent and to study it for various algorithms that are important in cryptography. In particular, we will compute the stability exponent exactly for exhaustive searches and for collision algorithms, and we will perform experiments to approximate the stability exponent of lattice reduction algorithms.

Remark 1. In practice, each actual problem might come equipped with one of many possible random inputs that is used to initialize the algorithm or to guide its intermediate stages. We will assume that these randomizing quantities are already included in the problem descriptions given by \mathcal{P} .

1. CRYPTOGRAPHIC STABILITY

We begin with a precise definition of the stability exponent of a sequence of algorithms.

Definition 1. Let \mathcal{A}_N be a sequence of algorithms and let $T_{\mathcal{A}_N}$ be the associated *running time* random variable. (The intuition is that \mathcal{A}_N is a particular algorithm applied to inputs of size approximately N .) The *normalized asymptotic running time distribution function* for the family of algorithms \mathcal{A} is defined to be

$$\tilde{F}_{\mathcal{A}}(t) = \lim_{N \rightarrow \infty} \text{Prob}(T_{\mathcal{A}_N} \leq E(T_{\mathcal{A}_N})t).$$

(Here $E(X)$ is the expectation the random variable X .) We denote by $\tilde{T}_{\mathcal{A}}$ a random variable whose distribution function is $\tilde{F}_{\mathcal{A}}$. In other words,

$$\text{Prob}(\tilde{T}_{\mathcal{A}} \leq t) = \tilde{F}_{\mathcal{A}}(t).$$

We will not concern ourselves with the precise conditions under which the function $\tilde{F}_{\mathcal{A}}$ and the random variable $\tilde{T}_{\mathcal{A}}$ exist. In practice, we will find that they exist for important classes of algorithms used in cryptanalysis.

Definition 2. Let \mathcal{A}_N be a sequence of algorithms as above. Let $\tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \dots$ be independent random variables each having the same distribution as $\tilde{T}_{\mathcal{A}}$. For every $K \geq 1$, let

$$\tilde{M}_K = \tilde{M}_{\mathcal{A},K} = \min\{\tilde{T}_1, \tilde{T}_2, \dots, \tilde{T}_K\}.$$

Thus if one performs K experiments, then \tilde{M}_K gives the (normalized) minimum running time. We define the (*Asymptotic*) *Stability Exponent* of the sequence of algorithms \mathcal{A}_N to be

$$\text{StExp}(\mathcal{A}) = \liminf_{K \rightarrow \infty} \frac{-\log E(\tilde{M}_K)}{\log K}.$$

We say that \mathcal{A} is *Cryptographically Stable* if

$$\text{StExp}(\mathcal{A}) \leq 1.$$

Remark 2. As described in the introduction, we can choose a cutoff time C and run the algorithm either until it terminates or until it has run for time C . We want to choose C to minimize the amount of time that it takes to solve at least one problem. The key quantity turns out to be the stability exponent, since

$$E(\min\{T_1, \dots, T_K\}) \approx \frac{E(T)}{K^{\text{StExp}}}.$$

So if we perform K experiments, we need to take

$$C \approx \frac{E(T)}{K^{\text{StExp}}}$$

to make it likely that we will get at least one successful conclusion. The total running time will be approximately KC , since all but one of the experiments will terminate at time C without having solved the problem. Hence the total running time is approximately

$$K \cdot \frac{E(T)}{K^{\text{StExp}}} = \frac{E(T)}{K^{\text{StExp}-1}}.$$

The conclusion we reach is:

- If $\text{StExp} \leq 1$, then using a time cutoff strategy does not reduce the amount of time required to solve one among many problems.
- If $\text{StExp} > 1$, then using a time cutoff strategy with cutoff $C = E(T)/K^{\text{StExp}}$ leads to a reduced time for solving one among K problems.

2. EXHAUSTIVE SEARCHES AND SYMMETRIC CIPHERS

Consider the problem of searching for the key to a symmetric cipher such as DES or AES from a search space Ω containing N keys. The best known algorithm tries each element of Ω until finding the right key. We denote this exhaustive search algorithm on N elements by $\mathcal{A}_N^{\text{exh}}$ and let $T_{\mathcal{A}_N^{\text{exh}}}$ be the corresponding running time. Then

$$\text{Prob}(T_{\mathcal{A}_N^{\text{exh}}} = i) = \frac{1}{N} \quad \text{for } 1 \leq i \leq N.$$

In other words, the running time for an exhaustive search on N objects is uniformly distributed on $\{1, \dots, N\}$.

The expected running time is $E(T_{\mathcal{A}_N^{\text{exh}}}) = (N + 1)/2$, and the normalized asymptotic running time distribution is

$$\begin{aligned}\tilde{F}(t) &= \lim_{N \rightarrow \infty} \text{Prob}(T_{\mathcal{A}_N^{\text{exh}}} \leq E(T_{\mathcal{A}_N^{\text{exh}}})t) \\ &= \lim_{N \rightarrow \infty} \text{Prob}(T_{\mathcal{A}_N^{\text{exh}}} \leq (N + 1)t/2) \\ &= \lim_{N \rightarrow \infty} \frac{1}{N} \left\lfloor \frac{(N + 1)t}{2} \right\rfloor \quad \text{for } 0 \leq t \leq \frac{2N}{N + 1} \\ &= \frac{t}{2} \quad \text{for } 0 \leq t \leq 2 \text{ and } 0 \text{ otherwise.}\end{aligned}$$

In other words, $\tilde{F}(t)$ is a uniform distribution function on the interval $[0, 2]$.

Proposition 1. *Let $\mathcal{A}_N^{\text{exh}}$ be the exhaustive search algorithm on a space of N objects. Then its normalized asymptotic running time \tilde{T}^{exh} is a uniform random variable on $[0, 2]$, and \mathcal{A}^{exh} is cryptographically stable with stability exponent*

$$\text{StExp}(\mathcal{A}^{\text{exh}}) = 1.$$

Proof. We have already proven that \tilde{T}^{exh} is uniform in $[0, 2]$. To ease notation, let $\tilde{T} = \tilde{T}^{\text{exh}}$, let $\tilde{T}_1, \tilde{T}_2, \dots$ be independent random variables that are uniformly distributed in $[0, 2]$ and let

$$\tilde{M}_K = \min\{\tilde{T}_1, \tilde{T}_2, \dots, \tilde{T}_K\}.$$

Then

$$\begin{aligned}\text{Prob}(\tilde{M}_K \geq t) &= \text{Prob}(\tilde{T}_i \geq t \text{ for all } i) \\ &= \text{Prob}(\tilde{T} \geq t)^K \quad \text{since } \tilde{T}_i \text{'s are independent,} \\ &= (1 - \text{Prob}(\tilde{T} \leq t))^K \\ &= \begin{cases} 1 & \text{if } t \leq 0, \\ (1 - t/2)^K & \text{if } 0 \leq t \leq 2, \\ 0 & \text{if } t \geq 2. \end{cases}\end{aligned}$$

Hence

$$E(\tilde{M}_K) = \int_0^\infty \text{Prob}(\tilde{M}_K \geq t) dt = \int_0^2 (1 - t/2)^K dt = \frac{2}{K + 1},$$

which enables us to compute

$$\lim_{K \rightarrow \infty} K \cdot E(\tilde{M}_K) = \lim_{K \rightarrow \infty} \frac{2K}{K + 1} = 2.$$

Thus $E(\tilde{M}_K) \sim 2K^{-1}$, which implies, and indeed is much stronger, than the assertion that

$$\text{StExp}(\tilde{M}) = \liminf_{K \rightarrow \infty} \frac{-\log E(\tilde{M}_K)}{\log K} = 1. \quad \square$$

3. COLLISION ALGORITHMS AND POLLARD'S RHO METHOD

In this section we consider the stability exponent of collision-type algorithms.

Proposition 2. *Let $\mathcal{A}_N^{\text{col}}$ denote a typical collision-type algorithm operating on a set of size N , for example Pollard's rho or lambda algorithm used to solve an elliptic curve discrete logarithm problem on an elliptic curve with approximately N points. Then $\mathcal{A}_N^{\text{col}}$ is cryptographically stable with stability exponent*

$$\text{StExp}(\mathcal{A}^{\text{col}}) = \frac{1}{2}.$$

Proof. Let T_N^{col} be running time associated to $\mathcal{A}_N^{\text{col}}$. The normalized running time of such an algorithm is given by the formula

$$\lim_{N \rightarrow \infty} \text{Prob}(T_N^{\text{col}} \leq t \cdot E(T_N^{\text{col}})) = 1 - e^{-\pi t^2/2} \quad \text{for } t \geq 0. \quad (1)$$

(See Appendix A for a derivation of this well-known result.) This formula is valid for $t \geq 0$, and clearly $\text{Prob}(T_N^{\text{col}} < 0) = 0$. If we take $\tilde{T}_1, \dots, \tilde{T}_K$ to be independent random variables with distribution functions given by (1) and set $\tilde{M}_K = \min\{\tilde{T}_1, \dots, \tilde{T}_K\}$, then we can compute

$$\begin{aligned} \text{Prob}(\tilde{M}_K \geq t) &= \text{Prob}(\tilde{T}_i \geq t \text{ for all } i) \\ &= \text{Prob}(\tilde{T} \geq t)^K \quad \text{since } \tilde{T}_i \text{'s are independent,} \\ &= (1 - \text{Prob}(\tilde{T} \leq t))^K \\ &= e^{-\pi K t^2/2} \quad \text{from (1).} \end{aligned}$$

Hence

$$\begin{aligned} E(\tilde{M}_K) &= \int_0^\infty \text{Prob}(\tilde{M}_K \geq t) dt = \int_0^\infty e^{-\pi K t^2/2} dt \\ &= \frac{1}{\sqrt{K}} \int_0^\infty e^{-\pi u^2/2} du. = \frac{1}{\sqrt{2K}} \end{aligned}$$

Therefore

$$\text{StExp}(\mathcal{A}^{\text{col}}) = \liminf_{K \rightarrow \infty} \frac{-\log E(\tilde{M}_K)}{\log K} = \frac{1}{2}. \quad \square$$

Dim	K	Mean	Min	StExp
170	100	334.0	151	0.172
180	100	449.0	205	0.170
190	100	673.9	281	0.190
200	100	1012.5	298	0.266
210	100	1562.4	522	0.238
220	100	2302.5	584	0.298
230	100	3569.2	1637	0.169
240	100	5521.8	1300	0.314
250	100	8994.0	2059	0.320
260	100	25213.1	5981	0.312
200	1000	1017.8	257	0.199

TABLE 1. Experimental Stability Exponent of LLL

4. LATTICE REDUCTION ALGORITHMS

The final type of algorithm that we will consider is lattice reduction, more specifically, the problem of finding the hidden short vector in a convolution modular lattice [1, 2, 3]. These lattice problems underlie the security of the NTRU cryptosystem. The best known algorithms for solving short(est) and close(est) vector problems are LLL and its variants. Unfortunately, there is very little known about the theoretical behavior of these algorithms. It is possible to prove upper bounds of various sorts, but in practice, the algorithms seem to do considerably better than predicted by these bounds.

We thus ran $K = 100$ experiments on NTRU lattices of various dimensions and recorded the time in seconds that it took BKZ-LLL (as implemented in NTL [4]) to solve each problem. We computed the mean and the minimum running times, and then we approximated the stability exponent using the formula

$$\text{StExp} \approx \frac{-\log\left(\frac{\text{Min Time}}{\text{Mean Time}}\right)}{\log K}.$$

The results of our experiments are recorded in Table 1, with the last column giving the approximation to StExp for the given experiments.

The last line of the table gives the results of extended experiments with lattices of dimension 200, for which we performed $K = 1000$ trials. We also computed an approximation to StExp using the 1000 experiments for the lattices of varying dimension. The result was a stability exponent of $\text{StExp} \approx 0.213$. It thus appears that the variation in running time of the BKZ-LLL lattice reduction algorithm applied to

convolution modular lattices is much too small to allow a cutoff strategy to be successfully employed.

APPENDIX A. RUNNING TIME FOR COLLISION ALGORITHMS

For the convenience of the reader, we derive the well-known running time estimate for collision algorithms. We model the algorithm by supposing that there are N boxes labeled $1, 2, \dots, N$ and that the algorithm randomly picks balls numbered between 1 and N and places them into the appropriate box. The algorithm successfully terminates when some box acquires two balls, i.e., when there is a collision. Let T_N^{col} be the associated running time. Then

$$\begin{aligned} \text{Prob}(T_N^{\text{col}} \leq M) &= 1 - \text{Prob}(T_N^{\text{col}} > M) \\ &= 1 - \text{Prob}(\text{no matches in } M \text{ trials}) \\ &= 1 - \prod_{m=1}^M \frac{N - (m - 1)}{N} \\ &= 1 - \prod_{m=0}^{M-1} \left(1 - \frac{m}{N}\right) \end{aligned}$$

This probability may be approximated by

$$\text{Prob}(T_N^{\text{col}} \leq M) \approx 1 - \prod_{m=0}^{M-1} e^{-m/N} \approx 1 - e^{-M^2/2N}. \quad (2)$$

Hence the expectation of T_N^{col} is given by

$$\begin{aligned} E(T_N^{\text{col}}) &\approx \int_0^\infty \text{Prob}(T_N^{\text{col}} \geq M) dM \\ &\approx \int_0^\infty e^{-M^2/2N} dM \\ &= c\sqrt{N} \quad \text{with } c = \int_0^\infty e^{-u^2/2} du = \sqrt{\frac{\pi}{2}}. \end{aligned}$$

This allows us to compute

$$\begin{aligned} \lim_{N \rightarrow \infty} \text{Prob}(T_N^{\text{col}} \leq t \cdot E(T_N^{\text{col}})) &= \lim_{N \rightarrow \infty} \text{Prob}(T_N^{\text{col}} \leq tc\sqrt{N}) \\ &= \lim_{N \rightarrow \infty} 1 - e^{-(tc\sqrt{N})^2/2N} \quad \text{from (2),} \\ &= 1 - e^{-t^2c^2}. \end{aligned}$$

Using the value $c = \sqrt{\pi/2}$, this is the formula (1) that we used in Section 3.

REFERENCES

- [1] J. Hoffstein, J. Pipher, J. Silverman, NTRU: A Ring Based Public Key Cryptosystem, Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, 267–288.
- [2] A. May, J.H. Silverman, Dimension reduction methods for convolution modular lattices, Conference on Lattices and Cryptography (CaLC 2001), Lecture Notes in Computer Science 2146, Springer-Verlag, 111–127.
- [3] NTRU Cryptosystems Technical Report #012: Estimated Breaking Times for NTRU Lattices, www.ntru.com/cryptolab/tech_notes.htm
- [4] V. Shoup, *NTL: A Number Theory Library* <www.shoup.net/ntl/>.

MATHEMATICS DEPARTMENT, BOX 1917, BROWN UNIVERSITY, PROVIDENCE,
RI 02912 USA AND NTRU CRYPTOSYSTEMS, INC., 5 BURLINGTON WOODS,
BURLINGTON, MA 01803

E-mail address: jhs@math.brown.edu