Non-uniform cracks in the concrete: the power of free precomputation

Daniel J. Bernstein[1] and Tanja Lange[2]

[1] Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7053, USA
djb@cr.yp.to
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
tanja@hyperelliptic.org

Abstract. There is a flaw in the standard security definitions used in the literature on provable concrete security. The definitions are frequently conjectured to assign a security level of 2^{128} to AES, the NIST P-256 elliptic curve, DSA-3072, RSA-3072, and various higher-level protocols, but they actually assign a far lower security level to each of these primitives and protocols. This flaw undermines security evaluations and comparisons throughout the literature. This paper analyzes the magnitude of the flaw in detail, showing how it varies across cryptosystems and across cost metrics, and analyzes several strategies for fixing the definitions.

Keywords: provable security, concrete security, non-uniform algorithms, algorithm cost metrics

1 Introduction

Why do we believe that AES-CBC-MAC is secure? More precisely: Why do we believe that an attacker limited to 2^{100} bit operations, and 2^{50} message blocks, cannot break AES-CBC-MAC with probability more than 2^{-20}?

The standard answer to this question has three parts. The first part is a concrete definition of what it means for a cipher or a MAC to be secure. We quote from the classic paper [8, Section 1.3] by Bellare, Kilian, and Rogaway: the PRP-"insecurity" of a cipher such as AES (denoted "$\mathbf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(q',t')$") is defined as the "maximum, over all adversaries restricted to q' input-output examples and execution time t', of the 'advantage' that the adversary has in the game of distinguishing [the cipher for a secret key] from a random permutation." The PRF-insecurity of m-block AES-CBC-MAC (denoted "$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CBC}^m\text{-AES}}(q,t)$") is defined similarly, using a uniform random function rather than a uniform random permutation.

The second part of the answer is a concrete security theorem bounding the insecurity of AES-CBC-MAC in terms of the insecurity of AES, or more generally the insecurity of F-CBC-MAC in terms of the insecurity of F for any ℓ-bit block cipher F. Specifically, here is the main theorem of [8]: "for any integers $q,t,m \ge 1$,

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CBC}^m\text{-}F}(q,t) \le \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t') + \frac{q^2m^2}{2^{l-1}}$$

where $q' = mq$ and $t' = t + O(mql)$." One can object that the O constant is unspecified, making this theorem meaningless as stated for any specific q,t,m values; but it is easy to imagine a truly concrete theorem replacing $O(mql)$ with the time for mql specified operations.

The third part of the answer is a concrete conjecture regarding the security of AES. NIST's call for AES submissions [59, Section 4] identified "the extent to which the algorithm output is indistinguishable from [the output of] a [uniform] random permutation" as one of the "most important" factors in evaluating candidates; cryptanalysts have extensively studied AES without finding any worrisome PRP-attacks; it seems reasonable to conjecture that no dramatically

This work was supported by the National Science Foundation under grant 1018836, and by the European Commission under Contract ICT-2007-216676 ECRYPT II. Permanent ID of this document: 7e044f2408c599254414615c72b3adbf. Date: 2013.03.05.

better attacks exist. Of course, this part of the story depends on the details of AES; analogous conjectures regarding, e.g., DES would have to be much weaker. For example, Bellare and Rogaway in [12, Section 3.6] wrote the following:

> "For example we might conjecture something like:
>
> $$\mathbf{Adv}^{\text{prp-cpa}}_{\text{DES}}(A_{t,q}) \le c_1 \cdot \frac{t/T_{\text{DES}}}{2^{55}} + c_2 \cdot \frac{q}{2^{40}}$$
>
> ... In other words, we are conjecturing that the best attacks are either exhaustive key search or linear cryptanalysis. We might be bolder with regard to AES and conjecture something like
>
> $$\mathbf{Adv}^{\text{prp-cpa}}_{\text{AES}}(B_{t,q}) \le c_1 \cdot \frac{t/T_{\text{AES}}}{2^{128}} + c_2 \cdot \frac{q}{2^{128}}.\text{"}$$

One can again object that the c_1 and c_2 are unspecified here, making these conjectures non-concrete and unfalsifiable as stated. A proper concrete conjecture would specify, e.g., $c_1 = c_2 = 3$. One can also quibble that the T_{DES} and T_{AES} factors do not properly account for inner-loop speedups in exhaustive key search (see, e.g., [22]), that $q/2^{40}$ is a rather crude model of the success probability of linear cryptanalysis, etc., but aside from such minor algorithm-analysis details the conjectures seem quite reasonable.

This AES security conjecture (with small specified c_1 and c_2) says, in particular, that the attacker cannot PRP-break AES with probability more than 2^{-21} after 2^{50} cipher outputs and 2^{100} bit operations. The CBC-MAC security theorem (with small specified O) then says that the same attacker cannot PRF-break AES-CBC-MAC with probability more than 2^{-20}.

Of course, this answer does not *prove* that AES-CBC-MAC is secure; it relies on a conjecture regarding AES security. Why not simply conjecture that AES-CBC-MAC is secure? The answer is scalability. It is reasonable to ask cryptanalysts to intensively study AES, eventually providing confidence in the security of AES, while it is much less reasonable to ask cryptanalysts to intensively study AES-CBC-MAC, AES-OMAC, AES-CCM, AES-GCM, AES-OCB, and hundreds of other AES-based protocols. Partitioning the AES-CBC-MAC security conjecture into an AES security conjecture and a CBC-MAC security proof drastically simplifies the cryptanalyst's job.

The same three-part pattern has (as illustrated by Appendix L) become completely standard throughout the literature on concrete "provable security". First part: The insecurity of X — where X is a primitive such as AES or RSA, or a higher-level protocol such as AES-CBC-MAC or RSA-PSS — is defined as the maximum, over all algorithms A ("attacks") that cost at most C, of the probability (or advantage in probability) that A succeeds in breaking X. This insecurity is explicitly a function of the cost limit C; typically C is separated into (1) a time limit t and (2) a limit q on the number of oracle queries. Note that this function depends implicitly on how the "cost" of an algorithm is defined.

Often "the (q, t)-insecurity of X is at most ϵ" is abbreviated "X is (q, t, ϵ)-secure". Many papers prefer the more concise notation and do not even mention the insecurity function. We emphasize, however, that this is merely a superficial change in notation, and that both of the quotes in this paragraph refer to exactly the same situation: namely, the nonexistence of algorithms that cost at most (q, t) and that break X with probability more than ϵ.

Second part: Concrete "provable security" theorems state that the insecurity (or security) of a complicated object is bounded in terms of the insecurity (or security) of a simpler object. Often these theorems require restrictions on the types of attacks allowed against the complicated object: for example, Bellare and Rogaway in [10] showed that RSA-OAEP has similar security to RSA against generic-hash attacks (attacks in the "random-oracle model").

Third part: The insecurity of a well-studied primitive such as AES or RSA-1024 is conjectured to match the success probability of the best attack known. For example, Bellare and Rogaway in [11, Section 1.4], evaluating the concrete-security of RSA-FDH and RSA-PSS, hypothesized that "it takes time $Ce^{1.923(\log N)^{1/3}(\log\log N)^{2/3}}$ to invert RSA"; Bellare in [5, Section 3.2], evaluating the concrete security of NMAC-h and HMAC-h, hypothesized that "the best attack against h as a PRF is exhaustive key search". These conjectures seem to precisely capture the idea that cryptanalysts will not make significant further progress in attacking these primitives.

1.1. Primary contribution of this paper. Our primary goal in this paper is to convincingly undermine all of the standard security conjectures reviewed above. Specifically, Sections 2, 3, 4, and 5 show — assuming standard, amply tested heuristics — that there *exist* high-probability attacks against AES, the NIST P-256 elliptic curve, DSA-3072, and RSA-3072 taking considerably less than 2^{128} time. In other words, the insecurity of AES, NIST P-256, DSA-3072, and RSA-3072, according to the standard concrete-security definitions, reaches essentially 100% for a time bound considerably below 2^{128}. The conjectures by Bellare and Rogaway in [**11**, Section 1.4], [**12**, Section 3.6], [**5**, Section 3.2], etc. are false for every reasonable assignment of the unspecified constants.

The same ideas show that there *exist* high-probability attacks against AES-CBC-MAC, RSA-3072-PSS, RSA-3072-OAEP, and thousands of other "provably secure" protocols, in each case taking considerably less than 2^{128} time. It is not clear that similar attacks exist against *every* such protocol in the literature, since in some cases the security reductions are unidirectional, but undermining these conjectures also means undermining all of the security arguments that have those conjectures as hypotheses.

We do not claim that this reflects any actual security problem with AES, NIST P-256, DSA-3072, and RSA-3072, or with higher-level protocols built from these primitives. On the contrary! Our constructions of these attacks are very slow; we conjecture that any *fast* construction of these attacks has negligible probability of success. Users have nothing to worry about.

However, the standard metrics count only the cost of running the attack, not the cost of finding the attack in the first place. This means that there is a very large gap between the actual insecurity of these primitives and their insecurity according to the standard metrics.

1.2. Secondary contribution of this paper. Our secondary goal in this paper is to propose a rescue strategy: a new way to define security — a definition that restores, to the maximum extent possible, the attractive three-part security arguments described above.

All of the gaps considered in this paper come from errors in quantifying feasibility. Each of the high-probability attacks presented in this paper (1) has a cost t according to the standard definitions, but (2) is obviously infeasible, even for an attacker able to carry out a "reasonable" algorithm that costs t according to the same definitions. The formalization challenge is to say exactly what "reasonable" means. Our core objective here is to give a new definition that accurately captures what is actually feasible for attackers.

This accuracy has two sides. First, the formally defined set of algorithms must be large enough. Security according to the definition does not imply actual security if the definition ignores algorithms that are actually feasible. Second, the formally defined set of algorithms must be small enough. One cannot conjecture security on the basis of cryptanalysis if infeasible attacks ignored by cryptanalysts are misdeclared to be feasible by the security definition.

We actually analyze four different ideas for modifying the notion of feasibility inside existing definitions: (1) switching the definitions from the RAM metric used in [**8**] to the NAND metric, an "alternative" mentioned in [**8**]; (2) switching the definitions to the AT metric, a standard hardware-design metric formally defined by Brent and Kung in [**24**] in 1981; (3) adding constructivity to the definitions, by a simple trick that we have not seen before (see Appendix B.4); and (4) adding uniformity to the definitions. Readers unfamiliar with the RAM, NAND, and AT metrics should see Appendix A for a summary and pointers to the literature.

Ultimately we recommend the second and third modifications as producing much more accurate models of actual feasibility. We also recommend refactoring theorems to simplify further changes, whether those changes are for even better accuracy or for other reasons. We recommend against the first and fourth modifications. Full details of our analysis appear in Appendix B; the NAND and AT analyses for individual algorithms appear in Sections 2, 3, 4, and 5. Appendix Q is a frequently-asked-questions list, serving a role for this paper comparable to the role that a traditional index serves for a book.

These two modifications have several positive consequences. Incorrect conjectures in the literature regarding the concrete security of primitives such as AES can be replaced by quite plausible conjectures using the new definitions. Our impression is that *most* of the proof ideas in the literature are compatible with the new definitions, modulo quantitative changes, so *most* concrete-security theorems in the literature can be replaced by meaningful concrete-security theorems using the new definitions.[3] The conjectures and theorems together will then produce reasonable conclusions regarding the concrete security of protocols such as AES-CBC-MAC.

1.3. Priority dates; credits; new analyses. On 20 March 2012 we publicly announced the trouble with the standard AES conjectures; on 17 April 2012 we publicly announced the trouble with the standard NIST P-256, DSA-3072, and RSA-3072 conjectures. The low-probability case of the AES trouble was observed independently by Koblitz and Menezes and announced earlier in March 2012; further credits to Koblitz and Menezes appear below. We are not aware of previous publications disputing the standard concrete-security conjectures.

Our attacks on AES, NIST P-256, DSA-3072, and RSA-3072 use many standard cryptanalytic techniques cited in Sections 2, 3, 4, and 5. We introduce new cost analyses in all four sections, and new algorithm improvements in Sections 3, 4, and 5; our improvements are critical for beating 2^{128} in Section 5. In Sections 2, 3, and 4 the standard techniques were already adequate to (heuristically) disprove the standard 2^{128} concrete-security conjectures, but as far as we know we were the first to point out these contradictions. We do not think the contradictions were obvious; in many cases the standard techniques were published decades *before* the conjectures!

This paper was triggered by a 23 February 2012 paper [**47**], in which Koblitz and Menezes objected to the non-constructive nature of Bellare's security proof [**5**] for NMAC. The security theorem states a quantitative relationship between the standard-definition-insecurity of NMAC-h and the standard-definition-insecurity of h: the *existence* of a fast attack on NMAC-h implies the *existence* of a fast attack on h. The objection is that the proof does not reveal a fast method to compute the second attack from the first: the proof left open the possibility that the fastest algorithm that can be *found* to attack NMAC-h is much faster than the fastest algorithm that can be *found* to attack h.

An early-March update of [**47**] added weight to this objection by pointing out the (heuristic) existence of a never-to-be-found fast algorithm to attack any 128-bit function h. The success probability of the algorithm was only about 2^{-64}, but this was still enough to disprove Bellare's security conjectures. Koblitz and Menezes commented on "how difficult it is to appreciate all the security implications of assuming that a function has prf-security even against unconstructible adversaries".

Compared to [**47**], we analyze a much wider range of attacks, including higher-probability PRF attacks and attacks against various public-key systems, showing that the difficulties here go far beyond PRF security. We also show quantitative variations of the difficulties between one algorithm cost metric and another, and we raise the possibility of eliminating the difficulties by carefully selecting a cost metric.

Readers who find these topics interesting may also be interested in the followup paper [**48**] by Koblitz and Menezes, especially the detailed discussion in [**48**, Section 2] of "two examples where the non-uniform model led researchers astray". See also Appendices Q.13, Q.14, and Q.15 of our paper for further comments on the concept of non-uniformity.

[3] We do not claim that *all* proofs can be rescued, and it is even possible that some theorems will have to be abandoned entirely. Some troublesome examples have been pointed out by Koblitz and Menezes in [**47**] and [**48**]. Our experience indicates, however, that such examples are unusual. For example, there is nothing troublesome about the CBC-MAC proof or the FDH proof; these proofs simply need to be placed in a proper framework of meaningful definitions, conjectures, and theorem statements.

2 Breaking AES

This section analyzes the cost of various attacks against AES. All of the attacks readily generalize to other block ciphers; none of the attacks exploit any particular weakness of AES. We focus on AES because of its relevance in practice and to have concrete numbers to illustrate the attacks.

All of the (single-target) attacks here are "PRP" attacks: i.e., attacks that distinguish the cipher outputs for a uniform random key (on attacker-selected inputs) from outputs of a uniform random permutation. Some of the attacks go further, recovering the cipher key, but this is not a requirement for a distinguishing attack.

2.1. Breaking AES with MD5. We begin with an attack that does not use any precomputations. This attack is feasible, and in fact quite efficient; its success probability is low, but not nearly as low as one might initially expect. This is a warmup for the higher-success-probability attack of Section 2.2.

Let P be a uniform random permutation of the set $\{0,1\}^{128}$; we label elements of this set in little-endian form as integers $0, 1, 2, \ldots$ without further comment. The pair $(P(0), P(1))$ is nearly a uniform random 256-bit string: it avoids 2^{128} strings of the form (x, x) but is uniformly distributed among the remaining $2^{256} - 2^{128}$ strings.

If k is a uniform random 128-bit string then the pair $(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$ is a highly nonuniform random 256-bit string, obviously incapable of covering more than 2^{128} possibilities. One can reasonably guess that an easy way to distinguish this string from $(P(0), P(1))$ is to feed it through MD5 and inspect the first bit of the result. The success probability of this attack — the absolute difference between its average output for input $(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$ and its average output for input $(P(0), P(1))$ — is far below 1, but it is almost certainly above 2^{-80}, and therefore many orders of magnitude above 2^{-128}. See Appendix V for relevant computer experiments.

To understand why this works, imagine replacing the first bit of MD5 with a uniform random function from $\{0,1\}^{256}$ to $\{0,1\}$, and assume for simplicity that the 2^{128} keys k produce 2^{128} distinct strings $(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$. Each key k then has a 50% chance of choosing 0 and a 50% chance of choosing 1, and these choices are independent, so the probability that $2^{127} + \delta$ keys k choose 1 is exactly $\binom{2^{128}}{2^{127}+\delta}/2^{2^{128}}$; the probability that *at least* $2^{127} + \delta$ keys k choose 1 is exactly $\sum_{i \geq \delta} \binom{2^{128}}{2^{127}+i}/2^{2^{128}}$; the probability that *at most* $2^{127} - \delta$ keys k choose 1 is the same. The other $2^{256} - 2^{129}$ possibilities for $(P(0), P(1))$ are practically guaranteed to have far smaller bias. This attack thus has success probability at least $\approx\delta/2^{128}$ with probability approximately $2\sum_{i \geq \delta} \binom{2^{128}}{2^{127}+i}/2^{2^{128}} \approx 1 - \mathrm{erf}(\delta/\sqrt{2^{127}}) \approx \exp(-\delta^2/2^{127})$, where erf is the standard error function. For example, the attack has success probability at least $\approx 2^{-65}$ with probability above 30%, and has success probability at least $\approx 2^{-80}$ with probability above 99.997%.

Of course, MD5 is not actually a uniform random function, but it would be astonishing for MD5 to interact with AES in such a way as to spoil this attack. More likely is that there are some collisions in $k \mapsto (\mathrm{AES}_k(0), \mathrm{AES}_k(1))$; but such collisions are rare unless AES is deeply flawed, and in any event will tend to push δ away from 0, helping the attack.

2.2. Precomputing larger success probabilities. The same analysis applies to a modified attack D_s that appends a short string s to the AES outputs $(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$ before hashing them: for each s, D_s has success probability at least $\approx\delta/2^{128}$ with probability $\approx \exp(-\delta^2/2^{127})$. If s is long enough to push the hash inputs beyond one block of MD5 input then the iterated structure of MD5 seems likely to spoil the attack, so we define D_s using "capacity-1024 Keccak" rather than MD5.

Consider, for example, $\delta = 2^{67}$: for each s, this attack D_s has success probability at least $\approx 2^{-61}$ with probability $\approx 1 - \mathrm{erf}(2^{3.5}) \approx 2^{-189}$. There are 2^{192} choices of 192-bit strings s, so presumably at least one of them will have D_s having success probability at least $\approx 2^{-61}$. Of course, actually *finding* such an s would require inconceivable amounts of computation by the best methods known (searching 2^{189} choices of s, and computing 2^{128} hashes for each choice);

but this is not relevant to the definition of insecurity, which considers only the time taken by D_s.

More generally, for any $n \in \{0, 1, 2, \ldots, 64\}$, D_s has success probability at least $\approx 2^{n-64}$ with probability $\approx 1 - \mathrm{erf}(2^{n+0.5}) \approx \exp(-2^{2n+1})$. There are $2^{3\cdot 2^{2n}}$ choices of $(3 \cdot 2^{2n})$-bit strings s, and $2^{3\cdot 2^{2n}}$ is considerably larger than $\exp(2^{2n+1})$, so presumably at least one of these values of s will have D_s having success probability at least $\approx 2^{n-64}$.

Similar comments apply to essentially any short-key cipher. There almost certainly *exists* a $(3 \cdot 2^{2n})$-bit string s such that the following simple attack achieves success probability $\approx 2^{n-K/2}$, where K is the number of bits in the cipher key: query $2K$ bits of cipher output, append s, and hash the result to 1 bit.

As n increases, the cost of hashing $3 \cdot 2^{2n} + 2K$ bits grows almost linearly with 2^{2n} in the RAM metric and the NAND metric. It grows more quickly in the AT metric: storing the $3 \cdot 2^{2n}$ bits of s uses area at least $3 \cdot 2^{2n}$, and even a heavily parallelizable hash function will take time proportional to 2^n simply to communicate across this area, for a total cost proportional to 2^{3n}. In each metric there are also lower-order terms reflecting the cost of hashing per bit; we suppress these lower-order terms since our concern is with much larger gaps.

2.3. Iteration. Large success probabilities are more efficiently achieved by a different type of attack that iterates, e.g., the function $f_7 : \{0,1\}^{128} \to \{0,1\}^{128}$ defined by $f_7(k) = \mathrm{AES}_k(0) \oplus 7$.

Choose an attack parameter n. Starting from $f_7(k)$, compute the sequence of iterates $f_7(k)$, $f_7^2(k), f_7^3(k), \ldots, f_7^{2^n}(k)$. Look up each of these iterates in a table containing the precomputed quantities $f_7^{2^n}(0), f_7^{2^n}(1), \ldots, f_7^{2^n}(2^n - 1)$. If $f_7^j(k)$ matches $f_7^{2^n}(i)$, recompute $f_7^{2^n-j}(i)$ as a guess for k, and verify this guess by checking $\mathrm{AES}_k(1)$.

This computation finds k if k matches any of the following keys: $0, f_7(0), \ldots, f_7^{2^n-1}(0)$; $1, f_7(1), \ldots, f_7^{2^n-1}(1)$; etc. If n is not too large (see the next paragraph) then there are close to 2^{2n} different keys here. The computation involves $\le 2^n$ initial iterations; 2^n table lookups; and, in case of a match, $\le 2^n$ iterations to recompute $f_7^{2^n-j}(i)$. The *precomputation* performs many more iterations, but this precomputation is only the cost of *finding* the algorithm, not the cost of *running* the algorithm.

This heuristic analysis begins to break down as $3n$ approaches the key size K. The central problem is that a chain $f_7(i), f_7^2(i), \ldots$ could collide with one of the other $2^n - 1$ chains; this occurs with probability $\approx 2^{3n}/2^K$, since there are 2^n keys in this chain and almost 2^{2n} keys in the other chains. The colliding chains will then merge, reducing the coverage of keys and at the same time requiring extra iterations to check more than one value of i. This phenomenon loses a small constant factor in the algorithm performance for $n \approx K/3$ and much more for larger n.

Assume from now on that n is chosen to be close to $K/3$. The algorithm then has success chance $\approx 2^{-K/3}$. The algorithm cost is on the scale of $2^{K/3}$ in both the RAM metric and the NAND metric; for the NAND metric one computes the 2^n independent table lookups by sorting and merging.

This attack might not sound better (in the RAM metric) than the earlier attack D_s, which achieves success chance $\approx 2^{-K/3}$ for some string s with $\approx 2^{K/3}$ bits. The critical advantage of this attack is that it recognizes its successes. If the attack fails to find k then one can change 7 to another number and try again, almost doubling the success chance of the algorithm at the expense of doubling its cost; for comparison, doubling the success chance of D_s requires quadrupling its cost. Repeating this attack $2^{K/3}$ times reaches success chance ≈ 1 at cost $2^{2K/3}$.

In the AT metric this attack is much more expensive. The table of precomputed quantities $f_7^{2^n}(0), f_7^{2^n}(1), \ldots, f_7^{2^n}(2^n - 1)$ uses area on the scale of 2^n, and computing $f_7^{2^n}(k)$ takes time on the scale of 2^n, for a total cost on the scale of 2^{2n} for an attack that finds $\approx 2^{2n}$ keys. One can *compute* $f_7^{2^n}(0), f_7^{2^n}(1), \ldots, f_7^{2^n}(2^n - 1)$ in parallel within essentially the same bounds on time and area, replacing each precomputed key with a small circuit that computes the key from scratch; precomputation does not change the exponent of the attack. One can, more straightforwardly, compute any reasonable sequence of 2^{2n} guesses for k within essentially the same cost bound. Achieving success probability p costs essentially $2^K p$.

2.4. Multiple targets. Iteration becomes more efficient when there are multiple targets: U cipher outputs $\mathrm{AES}_{k_1}(0), \mathrm{AES}_{k_2}(0), \ldots, \mathrm{AES}_{k_U}(0)$ for U independent uniform random keys $k_1, \ldots, k_U$. Assume for simplicity that U is much smaller than 2^K; the hypothesis $U \leq 2^{K/4}$ suffices for all heuristics used below.

Compute the iterates $f_7(k_1), f_7^2(k_1), \ldots, f_7^{2^n}(k_1)$, and similarly for each of $k_2, \ldots, k_U$; this takes $2^n U$ iterations. Look up each iterate in a table of $2^n U$ precomputed keys. Handle any match as above.

In the RAM metric or the NAND metric this attack has cost on the scale of $2^n U$, just like applying the previous attack to the U keys separately. The advantage of this attack is that it uses a larger table, producing a larger success probability for each key: the precomputation covers $2^{2n} U$ keys instead of just 2^{2n} keys. To avoid excessive chain collisions one must limit 2^n to $2^{K/3} U^{-1/3}$ so that $2^{3n} U$ does not grow past 2^K; the attack then finds each key with probability $2^{2n} U / 2^K = 2^{-K/3} U^{1/3}$, with a cost of $2^n = 2^{K/3} U^{-1/3}$ per key, a factor of $U^{2/3}$ better than handling each key separately. Finding each key with high probability costs $2^{2K/3} U^{-2/3}$ per key.

As before, the *AT* metric assigns a much larger cost than the RAM and NAND metrics. The computation of $f_7^{2^n}(k_1), f_7^{2^n}(k_2), \ldots, f_7^{2^n}(k_U)$ is trivially parallelized, taking time on the scale of 2^n, but the $2^n U$ precomputed keys occupy area $2^n U$, for a total cost on the scale of $2^{2n} U$, i.e., 2^{2n} per key, for success probability $2^{2n} U / 2^K$ per key. Note that one can carry out the precomputation using essentially the same area and time. There is a large benefit from handling U keys together — finding all U keys costs essentially 2^K, i.e., $2^K / U$ per key — but this benefit exists whether or not precomputation costs are taken into account.

2.5. Comparison. We summarize the insecurity established by the best attacks presented above. Achieving success probability p against U keys costs

- RAM metric: $\approx 2^K p^2$ for $p \leq 2^{-K/3} U^{-2/3}$; $\approx (2^{2K/3} / U^{2/3}) p$ for larger p.
- NAND metric: same.
- *AT* metric: $\approx 2^{3K/2} p^3$ for $p \leq 2^{-K/4} U^{-1/2}$; $\approx 2^K U^{-1} p$ for larger p.

Figure G.1 graphs these approximations for $U = 1$, along with the cost of exhaustive search.

2.6. Previous work. All of the attacks described here have appeared before. In fact, when the conjectures in [**12**, Section 3.6] and [**5**, Section 3.2] were made, they were already inconsistent with known attacks.

The iteration idea was introduced by Hellman in [**39**] for the special case $U = 1$. Many subsequent papers have explored variants and refinements of Hellman's attack, including the easy generalization to larger U. Hellman's goal was to attack many keys for a lower RAM cost than attacking each key separately; Hellman advertised a "cost per solution" of $2^{2K/3}$ using a precomputed table of size $2^{2K/3}$. The generalization to larger U achieves the same goal at lower cost, but the special case $U = 1$ remains of interest as a non-uniform single-key attack.

Koblitz and Menezes in [**47**] recently considered a family of attacks analogous to D_s. They explained that there should be a short string s where D_s has success probability at least $\approx 2^{-K/2}$, and analyzed some consequences for provable concrete secret-key security. However, they did not analyze higher levels of insecurity.

Replacing D_s with a more structured family of attacks, namely linear cryptanalysis, can be *proven* to achieve insecurity $2^{-K/2}$ at low cost. (See, for example, [**33**, Section 7], which says that this is "well known in complexity theory".) De, Trevisan, and Tulsiani in [**30**] proved cost $\approx 2^K p^2$, for both the RAM metric and the NAND metric, for any insecurity level p. A lucid discussion of the gap between these attacks and exhaustive search appears in [**30**, Section 1], but without any analysis of the resulting trouble for the literature on provable concrete secret-key security, and without any analysis of possible fixes.

Biham, Goren, and Ishai in [20, Section 1.1] pointed out that Hellman's attack causes problems for defining strong one-way functions. The only solution that they proposed was adding uniformity. Note that this solution abandons the goal of giving a definition for, e.g., the strength of AES as a one-way function, or the strength of protocols built on top of AES. We analyze this solution in detail in Appendix B.5.

Our AT analysis appears to be new. In particular, we are not aware of previous literature concluding that switching to the AT metric removes essentially all of the benefit of precomputation for large p, specifically $p > 2^{-K/4}U^{-1/2}$.

3 Breaking the NIST P-256 elliptic curve

This section analyzes the cost of an attack against NIST P-256 [60], an elliptic curve of 256-bit prime order ℓ over a 256-bit prime field $\mathbf{F}_p$. The attack computes discrete logarithms on this curve, recovering the secret key from the public key and thus completely breaking typical protocols that use NIST P-256.

The attack does not exploit any particular weakness of NIST P-256. Switching from NIST P-256 to another group of the same size (another curve over the same field, a curve over another field, a hyperelliptic curve, a torus, etc.) does not stop the attack. We focus on NIST P-256 for both concreteness and practical relevance, as in the previous section.

3.1. The standard attack without precomputation. Let P be the specified base point on the NIST P-256 curve. The discrete-logarithm problem on this curve is to find, given another point Q on this curve, the unique integer k modulo ℓ such that $Q = kP$. The standard attack against the discrete-logarithm problem is the parallelization by van Oorschot and Wiener [63] of Pollard's rho method [64], described in the following paragraphs.

This attack uses a pseudorandom walk on the curve points. To obtain the $(i+1)$-st point P_{i+1}, apply a hash function $h : \mathbf{F}_p \to I$ to the x-coordinate of P_i, select a step $S_{h(x(P_i))}$ from a sequence of precomputed steps $S_j = r_j P$ (with random scalars r_j for $j \in I$), and compute $P_{i+1} = P_i + S_{h(x(P_i))}$. The size of I is chosen large enough to have the walk simulate a uniform random walk; a common choice, recommended in [74], is $|I| = 20$. The walk continues until it hits a distinguished point: a point P_i where the last t bits of $x(P_i)$ are equal to zero. Here t is an attack parameter.

The starting point of the bth walk is of the form $aP + bQ$ where a is chosen randomly. Each step increases the multiple of P, so the distinguished point has the form $a'P + bQ$ for known a', b. The triple $(a'P + bQ, a', b)$ is stored and a new walk is started from a different starting point. If two walks hit the same distinguished point then $a'P + bQ = c'P + dQ$ which gives $(a' - c')P = (d - b)Q$; by construction $d \not\equiv b \bmod \ell$, revealing $k \equiv (a' - c')/(d - b) \bmod \ell$.

After $\sqrt{\ell} \approx 2^{128}$ additions (in approximately 2^{128-t} walks, using storage 2^{128-t}), there is a high chance that the same point has been obtained in two different walks. This collision is recognized from a repeated distinguished point within approximately 2^t additional steps.

3.2. Precomputed distinguished points. To use precomputations in this attack, build a database of triples of the form $(a'P, a', 0)$, i.e., starting each walk at a multiple of P. The attack algorithm takes this database and starts a new walk at $aP + bQ$ for random a and b. If this walk ends in a distinguished point present in the database, the DLP is solved. If the walk continues for more than 2^{t+1} steps (perhaps because it is in a cycle) or reaches a distinguished point not present in the database, the attack starts again from a new pair (a, b).

The parameter t is critical for RAM cost here, whereas it did not significantly affect RAM cost in Section 3.1. Choose t as $\lceil(\log_2 \ell)/3\rceil$. One can see from the following analysis that significantly smaller values of t are much less effective, and that significantly larger values of t are much more expensive without being much more effective.

Construct the database to have exactly 2^t distinct triples, each obtained from a walk of length at least 2^t, representing a total of at least 2^{2t} (and almost certainly $O(2^{2t})$) points. Achieving this requires searching for starting points in the precomputation (and optionally also varying the steps S_j and the hash function) as follows. A point that enters a cycle without reaching a distinguished point is discarded. A point that reaches a distinguished point in fewer than 2^t steps is discarded; each point survives this with probability approximately $(1-1/2^t)^{2^t} \approx 1/e$. A point that produces a distinguished point already in the database is discarded; to see that a point survives this with constant probability, observe that each new step has chance 2^{-t} of reaching a distinguished point, and chance $O(2^{2t}/\ell) = O(2^{-t})$ of reaching one of the previous $O(2^{2t})$ points represented by the database. Computer experiments that we reported in [**18**], as a followup to this paper, show that all the O constants here are reasonably close to 1.

Now consider a walk starting from $aP + bQ$. This walk has chance approximately $1/e$ of continuing for at least 2^t steps. If this occurs then those 2^t steps have chance approximately $1-(1-2^{2t}/\ell)^{2^t} \approx 1-\exp(-2^{3t}/\ell) \geq 1-1/e$ of reaching one of the 2^{2t} points in the precomputed walks that were within 2^t of the distinguished points in the database. If this occurs then the walk is guaranteed to reach a distinguished point in the database within a total of 2^{t+1} steps. The algorithm thus succeeds (in this way) with probability at least $(1-1/e)/e \approx 0.23$. This is actually an underestimate, since the algorithm can also succeed with an early distinguished point or a late collision.

To summarize, the attack uses a database of approximately $\sqrt[3]{\ell}$ distinguished points; one run of the attack uses approximately $2\sqrt[3]{\ell}$ curve additions and succeeds with rather high probability. The overall attack cost in the RAM metric is a small constant times $\sqrt[3]{\ell}$. The security of NIST P-256 in this metric has thus dropped to approximately 2^{86}. Note that the precomputation here is on the scale of 2^{170}, much larger than the precomputation in Section 2.3 but much smaller than the precomputation in Section 2.2.

In the NAND metric it is simplest to run each walk for exactly 2^{t+1} steps, keeping track of the first distinguished point found by that walk and then comparing that distinguished point to the 2^t points in the database. The overall attack cost is still on the scale of $\sqrt[3]{\ell}$.

In the AT metric the attack cost is proportional to $\sqrt[3]{\ell}^2$, larger than the standard $\sqrt{\ell}$. In this metric one does better by running many walks in parallel: if Z points are precomputed, one should run approximately Z walks in parallel with inputs depending on Q. The precomputation then covers $2^t Z$ points, and the computations involving Q cover approximately $2^t Z$ points, leading to a high probability of success when $2^t Z$ reaches $\sqrt{\ell}$. The AT cost is also $2^t Z$. This attack has the same cost as the standard Pollard rho method, except for small constants; there is no advantage in the precomputations.

3.3. Related work. Kuhn and Struik in [**50**] and Hitchcock, Montague, Carter, and Dawson in [**40**] considered the problem of solving multiple DLPs at once. They obtain a speedup of $\sqrt{U}$ per DLP for solving U DLPs at once. Their algorithm reuses the distinguished points found in the attack on Q_1 to attack Q_2, reuses the distinguished points found for Q_1 and Q_2 to attack Q_3, etc. However, their results do not seem to imply our $\sqrt[3]{\ell}$ result: they do not change the average walk length and distinguished-point probabilities, and they explicitly limit U to $c\sqrt[4]{\ell}$ with $c < 1$. See also the recent paper [**53**] by Lee, Cheon, and Hong, which considered solving DLPs with massive precomputation for trapdoor DL-groups. None of these papers noticed any implications for provable security, and none of them went beyond the RAM metric.

Our followup paper [**18**] experimentally verified the algorithm stated above, improved it to $1.77 \cdot \sqrt[3]{\ell}$ additions using $\sqrt[3]{\ell}$ distinguished points, extended it to DLPs in intervals (using slightly more additions), and showed constructive applications in various protocols.

4 Breaking DSA-3072

This section briefly analyzes the cost of an attack against the DSA-3072 signature system. The attack computes discrete logarithms in the DSA-3072 group, completely breaking the signature system.

DSA uses the unique order-q subgroup of the multiplicative group $\mathbf{F}_p^*$, where p and q are primes with q (and not q^2) dividing $p-1$. DSA-3072 uses a 3072-bit prime p and is claimed to achieve 2^{128} security. The standard parameter choices for DSA-3072 specify a 256-bit prime q, allowing the 2^{86} attack explained in Section 3, but this section assumes that the user has stopped this attack by increasing q to 384 bits (at a performance penalty).

4.1. The attack. Take $y = 2^{110}$, and precompute $\log_g x^{(p-1)/q}$ for every prime number $x \le y$, where g is the specified subgroup generator. There are almost exactly $y/\log y \approx 2^{103.75}$ such primes, and each $\log_g x^{(p-1)/q}$ fits into 48 bytes, for a total of $2^{109.33}$ bytes.

To compute $\log_g h$, first try to write h as a quotient h_1/h_2 in $\mathbf{F}_p^*$ with $h_2 \in \{1, 2, 3, \ldots, 2^{1535}\}$, $h_1 \in \{-2^{1535}, \ldots, 0, 1, \ldots, 2^{1535}\}$, and $\gcd\{h_1, h_2\} = 1$; and then try to factor h_1, h_2 into primes $\le y$. If this succeeds then $\log_g h^{(p-1)/q}$ is a known combination of known quantities $\log_g x^{(p-1)/q}$, revealing $\log_g h$. If this fails, try again with hg, hg^2, etc.

One can write h as h_1/h_2 with high probability, approximately $(6/\pi^2)2^{3071}/p$, since there are approximately $(6/\pi^2)2^{3071}$ pairs (h_1, h_2) and two distinct such pairs have distinct quotients. Finding the decomposition of h as h_1/h_2 is a very fast extended-Euclid computation.

The probability that h_1 is y-smooth (i.e., has no prime divisors larger than y) is very close to $u^{-u} \approx 2^{-53.06}$ where $u = 1535/110$. The same is true for h_2; overall the attack requires between $2^{107.85}$ and $2^{108.85}$ iterations, depending on $2^{3071}/p$. Batch trial division, analyzed in detail in Section 5, finds the y-smooth values among many choices of h_1 at very low cost in both the RAM metric and the NAND metric. This attack is much slower in the AT metric.

4.2. Previous work. Standard attacks against DSA-3072 do not rely on precomputation and cost more than 2^{128} in the RAM metric. These attacks have two stages: the first stage computes discrete logarithms of all primes $\le y$, and the second stage computes $\log_g h$. Normally y is chosen to minimize the cost of the first stage, whereas we replace the first stage by precomputation and choose y to minimize the cost of the second stage.

The simple algorithm reviewed here is not the state-of-the-art algorithm for the second stage; see, e.g., the "special-q descent" algorithms in [44] and [27]. The gap between known algorithms and existing algorithms is thus even larger than indicated in this section. We expect that re-optimizing these algorithms to minimize the cost of the second stage will produce even better results. We emphasize, however, that none of the algorithms perform well in the AT metric.

5 Breaking RSA-3072

This section analyzes the cost of an attack against RSA-3072. The attack completely breaks RSA-3072, factoring any given 3072-bit public key into its prime factors, so it also breaks protocols such as RSA-3072-FDH and RSA-3072-OAEP.

This section begins by stating a generalization of the attack to any RSA key size, and analyzing the asymptotic cost exponents of the generalized attack. It then analyzes the cost more precisely for 3072-bit keys.

5.1. NFS with precomputation. This attack is a variant of NFS, the standard attack against RSA. For simplicity this description omits several NFS optimizations. See [25] for an introduction to NFS.

The attack is determined by four parameters: a "polynomial degree" d; a "radix" m; a "height bound" H; and a "smoothness bound" y. Each of these parameters is a positive integer. The attack also includes a precomputed "factory"

$$F = \left\{ (a, b) \in \mathbf{Z} \times \mathbf{Z} : \begin{array}{l} -H \le a \le H;\ 0 < b \le H; \\ \gcd\{a, b\} = 1;\ \text{and } a - bm \text{ is } y\text{-smooth} \end{array} \right\}.$$

The standard estimate is that F has $(12/\pi^2)H^2/u^u$ elements where $u = (\log Hm)/\log y$. This estimate combines three approximations: first, there are about $12H^2/\pi^2$ pairs $(a, b) \in \mathbf{Z} \times \mathbf{Z}$

such that $-H \le a \le H$, $0 < b \le H$, and $\gcd\{a, b\} = 1$; second, $a - bm$ has approximately the same smoothness chance as a uniform random integer in $[1, Hm]$; third, the latter chance is approximately $1/u^u$.

The integers N factored by the attack will be between m^d and m^{d+1}. For example, with parameters $m = 2^{256}$, $d = 7$, $H = 2^{55}$, and $y = 2^{50}$, the attack factors integers between 2^{1792} and 2^{2048}. Parameter selection is analyzed later in more detail. The following three paragraphs explain how the attack handles N.

Write N in radix m: i.e., find $n_0, n_1, \ldots, n_d \in \{0, 1, \ldots, m-1\}$ such that $N = n_d m^d + n_{d-1} m^{d-1} + \cdots + n_0$. Compute the "set of relations"

$$R = \left\{(a, b) \in F : n_d a^d + n_{d-1} a^{d-1} b + \cdots + n_0 b^d \text{ is } y\text{-smooth}\right\}$$

using Bernstein's batch trial-division algorithm [15]. The standard estimate is that R has $(12/\pi^2)H^2/(u^u v^v)$ elements where $v = (\log((d+1)H^d m))/\log y$.

We pause the attack description to emphasize two important ways that this attack differs from conventional NFS: first, conventional NFS chooses m as a function of N, while this attack does not; second, conventional NFS computes R by sieving all pairs (a, b) with $-H \le a \le H$ and $0 < b \le H$ to detect smoothness of $a - bm$ and $n_d a^d + \cdots + n_0 b^d$ simultaneously, while this attack computes R by batch trial division of $n_d a^d + \cdots + n_0 b^d$ for the limited set of pairs $(a, b) \in F$.

The rest of the attack proceeds in the same way as conventional NFS. There is a standard construction of a sparse vector modulo 2 for each $(a, b) \in R$, and there is a standard way to convert several linear dependencies between the vectors into several congruences of squares modulo N, producing the complete prime factorization of N; see [25] for details. The number of components of each vector is approximately $2y/\log y$, and standard sparse-matrix techniques find linear dependencies using about $4y/\log y$ simple operations on dense vectors of length $2y/\log y$. If the number of elements of R is larger than the number of components of each vector then linear dependencies are guaranteed to exist.

5.2. Asymptotic exponents. Write $L = \exp((\log N)^{1/3}(\log\log N)^{2/3})$. For the RAM metric it is best to choose

$$\begin{aligned}
d &\in (1.1047\ldots + o(1))(\log N)^{1/3}(\log\log N)^{-1/3},\\
\log m &\in (0.9051\ldots + o(1))(\log N)^{2/3}(\log\log N)^{1/3},\\
\log y &\in (0.8193\ldots + o(1))(\log N)^{1/3}(\log\log N)^{2/3} = (0.8193\ldots + o(1))\log L,\\
\log H &\in (1.0034\ldots + o(1))(\log N)^{1/3}(\log\log N)^{2/3} = (1.0034\ldots + o(1))\log L.
\end{aligned}$$

so that

$$\begin{aligned}
u &\in (1.1047\ldots + o(1))(\log N)^{1/3}(\log\log N)^{-1/3}, u\log u = (0.3682\ldots + o(1))\log L,\\
d\log H &\in (1.1085\ldots + o(1))(\log N)^{2/3}(\log\log N)^{1/3},\\
v &\in (2.4578\ldots + o(1))(\log N)^{1/3}(\log\log N)^{-1/3}, v\log v = (0.8193\ldots + o(1))\log L.
\end{aligned}$$

Out of the $L^{2.0068\ldots+o(1)}$ pairs (a, b) with $-H \le a \le H$ and $0 < b \le H$, there are $L^{1.6385\ldots+o(1)}$ pairs in the factory F, and $L^{0.8193\ldots+o(1)}$ relations in R, just enough to produce linear dependencies if the $o(1)$ terms are chosen appropriately. Linear algebra uses $y^{2+o(1)} = L^{1.6385\ldots+o(1)}$ bit operations.

The total RAM cost of this factorization algorithm is thus $L^{1.6385\ldots+o(1)}$. For comparison, factorization is normally claimed to cost $L^{1.9018\ldots+o(1)}$ (in the RAM metric) with state-of-the-art variants of NFS. Similar comments apply to the NAND metric.

This algorithm runs into trouble in the AT metric. The algorithm needs space to store all the elements of F, and can compute R in time $L^{o(1)}$ using a chip of that size (applying ECM to each input in parallel rather than using batch trial division), but even the most heavily parallelized sparse-matrix techniques need much more than $L^{o(1)}$ time, raising the AT cost of the algorithm far above the size of F. A quantitative analysis shows that one obtains a better cost exponent by skipping the precomputation of F and instead computing the elements of F one by one on a smaller circuit, for AT cost $L^{1.9760...+o(1)}$.

5.3. RAM cost for RSA-3072. This attack breaks RSA-3072 with RAM cost considerably below the 2^{128} security level usually claimed for RSA-3072. Of course, justifying this estimate requires replacing the above $o(1)$ terms with more precise cost analyses.

For concreteness, assume that the RAM supports 128-bit pointers, unit-cost 256-bit vector operations, and unit-cost 256-bit floating-point multiplications. As justification for these assumptions, observe that real computers ten years ago supported 32-bit pointers, unit-cost 64-bit vector operations, and unit-cost 64-bit floating-point multiplications; that the RAM model requires operations to scale logarithmically with the machine size; and that previous NFS cost analyses implicitly make similar assumptions.

Take $m = 2^{384}$, $d = 7$, $H = 2^{62}+2^{61}+2^{57}$, and $y = 2^{66}+2^{65}$. There are about $12H^2/\pi^2 \approx 2^{125.51}$ pairs (a, b) with $-H \le a \le H$, $0 < b \le H$, and $\gcd\{a, b\} = 1$, and the integers $a - bm$ have smoothness chance approximately $u^{-u} \approx 2^{-18.42}$ where $u = (\log Hm)/\log y \approx 6.707$, so there are about $2^{107.09}$ pairs in the factory F. Each pair in F is small, easily encoded as just 16 bytes.

The quantities $n_d a^d + n_{d-1}a^{d-1}b + \cdots + n_0 b^d$ are bounded by $(d+1)mH^d \approx 2^{825.3}$. If they were uniformly distributed up to this bound then they would have smoothness chance approximately $v^{-v} \approx 2^{-45.01}$ where $v = (\log((d+1)mH^d))/\log y \approx 12.395$, so there would be approximately $(12H^2/\pi^2)u^{-u}v^{-v} \approx 2^{62.08}$ relations, safely above $2y/\log y \approx 2^{62.06}$. The quantities $n_d a^d + n_{d-1}a^{d-1}b + \cdots + n_0 b^d$ are actually biased towards smaller values and thus have larger smoothness chance, but this refinement is unnecessary here.

Batch trial division checks smoothness of 2^{58} of these quantities simultaneously; here 2^{58} is chosen so that the product of those quantities is larger (about $2^{67.69}$ bits) than the product of all the primes $\le y$ (about $2^{67.11}$ bits). The main steps in batch trial division are computing a product tree of these quantities and then computing a scaled remainder tree. Bernstein's cost analysis in [**16**, Section 3] shows that the overall cost of these two steps, for T inputs having a B-bit product, is approximately $(5/6)\log_2 T$ times the cost of a single multiplication of two $(B/2)$-bit integers. For us $T = 2^{58}$ and $B \approx 2^{67.69}$, and the cost of batch trial division is approximately $2^{5.59}$ times the cost of multiplying two $(B/2)$-bit integers; the total cost of smoothness detection for all $(a, b) \in F$ is approximately $2^{54.68}$ times the cost of multiplying two $(B/2)$-bit integers.

It is easiest to follow a standard floating-point multiplication strategy, dividing each $(B/2)$-bit input into $B/(2w)$ words for some word size $w \in \Omega(\log_2 B)$ and then performing three real floating-point FFTs of length B/w. Each FFT uses approximately $(17/9)(B/w)\log_2(B/w)$ arithmetic operations (additions, subtractions, and multiplications) on words of slightly more than $2w$ bits, for a total of $(17/3)(B/w)\log_2(B/w)$ arithmetic operations. A classic observation of Schönhage [**70**] is that the RAM metric allows constant-time multiplication of $\Theta(\log_2 B)$-bit integers in this context even if the machine model is not assumed to be equipped with a multiplier, since one can afford to build large multiplication tables; but it is simpler to take advantage of the hypothesized 256-bit multiplier, which comfortably allows $w = 69$ and $B/w < 2^{61} + 2^{60}$, for a total multiplication cost of $2^{70.03}$. Computing R then costs approximately $2^{124.71}$.

Linear algebra involves $2^{63.06}$ simple operations on vectors of length $2^{62.06}$. Each operation produces each output bit by xoring together a small number of input bits, on average fewer than 32 bits. A standard block-Wiedemann computation merges 256 xors of bits into a single 256-bit xor with negligible overhead, for a total linear-algebra cost of $2^{122.12}$. All other steps in the algorithm have negligible cost, so the final factorization cost is $2^{124.93}$.

5.4. Previous work. There are two frequently quoted cost exponents for NFS without precomputation. Buhler, Lenstra, and Pomerance in [**25**] obtained RAM cost $L^{1.9229...+o(1)}$. Coppersmith in [**28**] introduced a "multiple number fields" tweak and obtained RAM cost $L^{1.9018...+o(1)}$.

Coppersmith also introduced NFS with precomputation in [**28**], using ECM for smoothness detection. Coppersmith called his algorithm a "factorization factory", emphasizing the distinction between precomputation time (building the factory) and computation time (running the factory). Coppersmith computed the same RAM exponent 1.6385 . . . shown above for the cost of one factorization using the factory.

We save a subexponential factor in the RAM cost of Coppersmith's algorithm by switching from ECM to batch trial division. This is not visible in the asymptotic exponent 1.6385 . . . but is important for RSA-3072. Our concrete analysis of RSA-3072 security is new, and as far as we know is the *first concrete analysis of Coppersmith's algorithm*.

Bernstein in [**14**] obtained *AT* exponent 1.9760 . . . for NFS without precomputation, and emphasized the gap between this exponent and the RAM exponent 1.9018 . . .. Our *AT* analysis of NFS with precomputation, and in particular our conclusion that this precomputation increases the *AT* cost of NFS, appears to be new.

References

[1] — (no editor), *38th annual symposium on coundations of computer science, FOCS '97, Miami Beach, Florida, USA, October 19–22, 1997*, IEEE Computer Society, 1997. See [6].

[2] Scott Aaronson, *Lecture 20: cosmology and complexity* (2006). URL: http://www.scottaaronson.com/democritus/lec20.html. Citations in this document: §Q.10.

[3] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, *The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory*, Journal of Computer and System Sciences **77** (2011), 14–40. Citations in this document: §B.4.

[4] Jacob D. Bekenstein, *Universal upper bound on the entropy-to-energy ratio for bounded systems*, Physical Review D **23** (1981), 287–298. Citations in this document: §Q.10.

[5] Mihir Bellare, *New proofs for NMAC and HMAC: security without collision-resistance*, in Crypto 2006 [**34**] (2006), 602–619. URL: http://cseweb.ucsd.edu/~mihir/papers/hmac-new.html. Citations in this document: §1, §1.1, §1.3, §2.6, §B.1, §Q.18.

[6] Mihir Bellare, Anand Desai, Eron Jokipii, Phillip Rogaway, *A concrete security treatment of symmetric encryption*, in [**1**] (1997), 394–403. URL: http://cseweb.ucsd.edu/~mihir/papers/sym-enc.html. Citations in this document: §Q.19.

[7] Mihir Bellare, Joe Kilian, Phillip Rogaway, *The security of cipher block chaining*, in Crypto 1994 [**32**] (1994), 341–358; see also newer version [**8**]. Citations in this document: §A.1.

[8] Mihir Bellare, Joe Kilian, Phillip Rogaway, *The security of the cipher block chaining message authentication code*, Journal of Computer and System Sciences **61** (2000), 362–399; see also older version [**7**]. ISSN 0022–0000. URL: http://www-cse.ucsd.edu/~mihir/papers/cbc.html. Citations in this document: §1, §1, §1.2, §1.2, §A.1, §A.1, §A.1, §A.1, §A.2, §B.2, §B.5, §B.6, §B.6.

[9] Mihir Bellare, Thomas Ristenpart, Stefano Tessaro, *Multi-instance security and its application to password-based cryptography*, in Crypto 2012 [**69**] (2012), 312–329. Citations in this document: §L.1, §L.1.

[10] Mihir Bellare, Phillip Rogaway, *Optimal asymmetric encryption — how to encrypt with RSA*, in Eurocrypt 1994 [**31**] (1995), 92–111. URL: http://cseweb.ucsd.edu/~mihir/papers/oaep.html. Citations in this document: §1.

[11] Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin*, in Eurocrypt 1996 [**56**] (1996), 399–416. URL: http://www-cse.ucsd.edu/~mihir/papers/exactsigs.html. Citations in this document: §1, §1.1, §B.1, §Q.18.

[12] Mihir Bellare, Phillip Rogaway, *Introduction to modern cryptography*, 2005. URL: http://cseweb.ucsd.edu/~mihir/cse207/classnotes.html. Citations in this document: §1, §1.1, §2.6, §B.1, §Q.18.

[13] Daniel J. Bernstein, *How to stretch random functions: the security of protected counter sums*, Journal of Cryptology **12** (1999), 185–192. URL: http://cr.yp.to/papers.html#stretch. Citations in this document: §B.6.

[14] Daniel J. Bernstein, *Circuits for integer factorization: a proposal* (2001). URL: http://cr.yp.to/papers.html#nfscircuit. Citations in this document: §5.4, §A.3, §Q.8.

[15] Daniel J. Bernstein, *How to find smooth parts of integers* (2004). URL: http://cr.yp.to/papers.html#smoothparts. Citations in this document: §5.1.

[16] Daniel J. Bernstein, *Scaled remainder trees* (2004). URL: http://cr.yp.to/papers.html#scaledmod. Citations in this document: §5.3.

[17] Daniel J. Bernstein, *Proving tight security for Rabin–Williams signatures*, in Eurocrypt 2008 [**72**] (2008), 70–87. URL: http://cr.yp.to/papers.html#rwtight. Citations in this document: §B.6.

[18] Daniel J. Bernstein, Tanja Lange, *Computing small discrete logarithms faster*, in Indocrypt 2012 [**35**] (2012), 317–338. URL: http://cr.yp.to/papers.html#cuberoot. Citations in this document: §3.2, §3.3.

[19] Gianfranco Bilardi, Franco P. Preparata, *Horizons of parallel computation*, Journal of Parallel and Distributed Computing **27** (1995), 172–182. Citations in this document: §A.3, §Q.6.

[20] Eli Biham, Yaron J. Goren, Yuval Ishai, *Basing weak public-key cryptography on strong one-way functions*, in TCC 2008 [**26**] (2008), 55–72. Citations in this document: §2.6.

[21] Peter van Emde Boas, *Machine models and simulation*, in [**54**] (1990), 1–66. Citations in this document: §A.1.

[22] Andrey Bogdanov, Dmitry Khovratovich, Christian Rechberger, *Biclique cryptanalysis of the full AES*, in Asiacrypt 2011 [**52**] (2011), 344–371. URL: `http://eprint.iacr.org/2011/449`. Citations in this document: §1.

[23] Raphel Bousso, *Positive vacuum energy and the N-bound*, Journal of High Energy Physics **11** (2000), 038. URL: `http://arxiv.org/abs/hep-th/0010252`. Citations in this document: §Q.10.

[24] Richard P. Brent, H. T. Kung, *The area-time complexity of binary multiplication*, Journal of the ACM **28** (1981), 521–534. URL: `http://wwwmaths.anu.edu.au/~brent/pub/pub055.html`. Citations in this document: §1.2, §A.3, §A.3, §Q.8, §Q.25.

[25] Joe P. Buhler, Hendrik W. Lenstra, Jr., Carl Pomerance, *Factoring integers with the number field sieve*, in [**55**] (1993), 50–94. Citations in this document: §5.1, §5.1, §5.4.

[26] Ran Canetti (editor), *Theory of cryptography, fifth theory of cryptography conference, TCC 2008, New York, USA, March 19–21, 2008*, Lecture Notes in Computer Science, 4948, Springer, 2008. ISBN 978-3-540-78523-1. See [20].

[27] An Commeine, Igor Semaev, *An algorithm to solve the discrete logarithm problem with the number field sieve*, in PKC 2006 [**77**] (2006), 174–190. Citations in this document: §4.2.

[28] Don Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology **6** (1993), 169–180. Citations in this document: §5.4, §5.4.

[29] Anindya De, Luca Trevisan, Madhur Tulsiani, *Non-uniform attacks against one-way functions and PRGs*, Electronic Colloquium on Computational Complexity **113** (2009); see also newer version [**30**].

[30] Anindya De, Luca Trevisan, Madhur Tulsiani, *Time space tradeoffs for attacks against one-way functions and PRGs*, in Crypto 2010 [**66**] (2010), 649–665; see also older version [**29**]. Citations in this document: §2.6, §2.6.

[31] Alfredo De Santis (editor), *Advances in cryptology — EUROCRYPT '94, workshop on the theory and application of cryptographic techniques, Perugia, Italy, May 9–12, 1994, proceedings*, Lecture Notes in Computer Science, 950, Springer, 1995. ISBN 3-540-60176-7. MR 98h:94001. See [10].

[32] Yvo Desmedt (editor), *Advances in cryptology — CRYPTO '94, 14th annual international cryptology conference, Santa Barbara, California, USA, August 21–25, 1994, proceedings*, Lecture Notes in Computer Science, 839, Springer, 1994. ISBN 3-540-58333-5. See [7].

[33] Yevgeniy Dodis, John Steinberger, *Message authentication codes from unpredictable block ciphers*, in Crypto 2009 [**38**] (2009), 267–285. URL: `http://cs.nyu.edu/~dodis/ps/tight-mac.pdf`. Citations in this document: §2.6.

[34] Cynthia Dwork (editor), *Advances in cryptology — CRYPTO 2006, 26th annual international cryptology conference, Santa Barbara, California, USA, August 20–24, 2006, proceedings*, Lecture Notes in Computer Science, 4117, Springer, 2006. ISBN 3-540-37432-9. See [5].

[35] Steven Galbraith, Mridul Nandi (editors), *Progress in cryptology — Indocrypt 2012 — 13th international conference on cryptology in India, Kolkata, India, December 9–12, 2012, proceedings*, Lecture Notes in Computer Science, 7668, Springer, 2012. See [18].

[36] Martin Gardner, *Mathematical games: the fantastic combinations of John Conway's new solitaire game "life"*, Scientific American **223** (1970), 120–123. URL: `http://web.archive.org/web/20090603015231/http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm`. Citations in this document: §B.4.

[37] Oded Goldreich, *P, NP, and NP-completeness: the basics of computational complexity*, 2009. URL: `http://www.wisdom.weizmann.ac.il/~oded/CC/bc-3.ps`. Citations in this document: §Q.15.

[38] Shai Halevi (editor), *Advances in cryptology — CRYPTO 2009, 29th annual international cryptology conference, Santa Barbara, CA, USA, August 16–20, 2009, proceedings*, Lecture Notes in Computer Science, 5677, Springer, 2009. See [33].

[39] Martin E. Hellman, *A cryptanalytic time-memory tradeoff*, IEEE Transactions on Information Theory **26** (1980), 401–406. Citations in this document: §2.6.

[40] Yvonne Hitchcock, Paul Montague, Gary Carter, Ed Dawson, *The efficiency of solving multiple discrete logarithm problems and the implications for the security of fixed elliptic curves*, International Journal of Information Security **3** (2004), 86–98. Citations in this document: §3.3.

[41] Viet Tung Hoang, Ben Morris, Phillip Rogaway, *An enciphering scheme based on a card shuffle*, in Crypto 2012 [**69**] (2012), 1–13. Citations in this document: §L.1, §L.1.

[42] Dennis Hofheinz, Tibor Jager, *Tightly secure signatures and public-key encryption*, in Crypto 2012 [**69**] (2012), 590–607. Citations in this document: §L.1, §L.1.

[43] Tibor Jager, Florian Kohlar, Sven Schäge, Jörg Schwenk, *On the security of TLS-DHE in the standard model*, in Crypto 2012 [**69**] (2012), 273–293. Citations in this document: §L.1, §L.1, §L.1.

[44] Antoine Joux, Reynald Lercier, *Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method*, Mathematics of Computation **72** (2003), 953–967. Citations in this document: §4.2.
[45] Jonathan Katz, Yehuda Lindell, *Introduction to modern cryptography: principles and protocols*, Chapman & Hall/CRC, 2007. ISBN 978-1-58488-551-1. URL: http://www.cs.umd.edu/~jkatz/imc.html. Citations in this document: §L.
[46] Jonathan Katz, *Non-uniform complexity* (2011). URL: http://www.cs.umd.edu/~jkatz/complexity/f11/lecture11.pdf. Citations in this document: §Q.15.
[47] Neal Koblitz, Alfred Menezes, *Another look at HMAC* (2012). URL: http://eprint.iacr.org/2012/074. Citations in this document: §3, §1.3, §1.3, §1.3, §2.6, §Q.22.
[48] Neal Koblitz, Alfred Menezes, *Another look at non-uniformity* (2012). URL: http://eprint.iacr.org/2012/359. Citations in this document: §3, §1.3, §1.3.
[49] Pascal Koiran, *Decision versus evaluation in algebraic complexity* (2007). URL: http://perso.ens-lyon.fr/pascal.koiran/Publis/mcu07.pdf. Citations in this document: §Q.15.
[50] Fabian Kuhn, Rene Struik, *Random walks revisited: extensions of Pollard's rho algorithm for computing multiple discrete logarithms*, in SAC 2001 [**75**] (2001), 212–229. URL: http://www.distcomp.ethz.ch/publications.html. Citations in this document: §3.3.
[51] Will Landecker, Thomas Shrimpton, R. Seth Terashima, *Tweakable blockciphers with beyond birthday-bound security*, in Crypto 2012 [**69**] (2012), 14–30. Citations in this document: §L.1, §L.1.
[52] Dong Hoon Lee, Xiaoyun Wang (editors), *Advances in cryptology — ASIACRYPT 2011, 17th international conference on the theory and application of cryptology and information security, Seoul, South Korea, December 4–8, 2011, proceedings*, Lecture Notes in Computer Science, 7073, Springer, 2011. ISBN 978-3-642-25384-3. See [22].
[53] Hyung Tae Lee, Jung Hee Cheon, Jin Hong, *Accelerating ID-based encryption based on trapdoor DL using pre-computation*, 11 Jan 2012 (2012). URL: http://eprint.iacr.org/2011/187. Citations in this document: §3.3.
[54] Jan van Leeuwen (editor), *Handbook of theoretical computer science, volume A: algorithms and complexity*, MIT Press, 1990. ISBN 0-262-22038-5. See [21].
[55] Arjen K. Lenstra, Hendrik W. Lenstra, Jr. (editors), *The development of the number field sieve*, Lecture Notes in Mathematics, 1554, Springer-Verlag, 1993. ISBN 3-540-57013-6. MR 96m:11116. See [25].
[56] Ueli M. Maurer (editor), *Advances in cryptology — EUROCRYPT '96: proceedings of the fifteenth international conference on the theory and application of cryptographic techniques held in Saragossa, May 12–16, 1996*, Lecture Notes in Computer Science, 1070, Springer, 1996. ISBN 3-540-61186-X. MR 97g:94002. See [11].
[57] Alfred Menezes, Edlyn Teske, Annegret Weng, *Weak fields for ECC*, in CT-RSA 2004 [**62**] (2004), 366–386. URL: http://eprint.iacr.org/2003/128. Citations in this document: §B.1, §B.1.
[58] Chris Nyberg, Mehul Shah, *Sort benchmark home page*, updated 23 May 2012, accessed 11 February 2013 (2012). URL: http://sortbenchmark.org. Citations in this document: §Q.6.
[59] National Institute for Standards and Technology, *Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES)* (1997). URL: http://www.gpo.gov/fdsys/pkg/FR-1997-09-12/pdf/97-24214.pdf. Citations in this document: §1.
[60] National Institute for Standards and Technology, *Digital signature standard*, Federal Information Processing Standards Publication 186-2 (2000). URL: http://csrc.nist.gov. Citations in this document: §3.
[61] Phong Q. Nguyen (editor), *Progress in cryptology — VIETCRYPT 2006, first international conference on cryptology in Vietnam, Hanoi, Vietnam, September 25–28, 2006, revised selected papers*, Lecture Notes in Computer Science, 4341, Springer, 2006. ISBN 3-540-68799-8. See [67].
[62] Tatsuaki Okamoto, *Topics in cryptology — CT-RSA 2004, the cryptographers' track at the RSA Conference 2004, San Francisco, CA, USA, February 23–27, 2004, proceedings*, Lecture Notes in Computer Science, 2964, Springer, 2004. ISBN 3-540-20996-4. See [57].
[63] Paul C. van Oorschot, Michael Wiener, *Parallel collision search with cryptanalytic applications*, Journal of Cryptology **12** (1999), 1–28. ISSN 0933–2790. URL: http://members.rogers.com/paulv/papers/pubs.html. Citations in this document: §3.1.
[64] John M. Pollard, *Monte Carlo methods for index computation mod p*, Mathematics of Computation **32** (1978), 918–924. ISSN 0025–5718. MR 58:10684. URL: http://www.ams.org/journals/mcom/1978-32-143/S0025-5718-1978-0491431-9/S0025-5718-1978-0491431-9.pdf. Citations in this document: §3.1.
[65] Franco P. Preparata, *Optimal three-dimensional VLSI layouts*, Mathematical Systems Theory **16** (1983), 1–8. Citations in this document: §A, §Q.8.
[66] Tal Rabin (editor), *Advances in cryptology — CRYPTO 2010, 30th annual cryptology conference, Santa Barbara, CA, USA, August 15–19, 2010, proceedings*, Lecture Notes in Computer Science, 6223, Springer, 2010. See [30].
[67] Phillip Rogaway, *Formalizing human ignorance*, in VIETCRYPT 2006 [**61**] (2006), 211–228. URL: http://www.cs.ucdavis.edu/~rogaway/papers/. Citations in this document: §B.5, §B.6, §Q.18, §Q.20.
[68] Arnold L. Rosenberg, *Three-dimensional VLSI: a case study*, Journal of the ACM **30** (1983), 397–416. Citations in this document: §A, §Q.8.

[69] Reihaneh Safavi-Naini, Ran Canetti (editors), *Advances in cryptology — CRYPTO 2012 — 32nd annual cryptology conference, Santa Barbara, CA, USA, August 19–23, 2012, proceedings*, Lecture Notes in Computer Science, 7417, Springer, 2012. ISBN 978-3-642-32008-8. See [9], [41], [42], [43], [51].

[70] Arnold Schönhage, *Storage modification machines*, SIAM Journal on Computing **9** (1980), 490–508. Citations in this document: §5.3.

[71] Peter W. Shor, *Introduction to quantum algorithms* (2001). URL: http://arxiv.org/abs/quant-ph/0005003. Citations in this document: §Q.15.

[72] Nigel P. Smart (editor), *Advances in cryptology — EUROCRYPT 2008, 27th annual international conference on the theory and applications of cryptographic techniques, Istanbul, Turkey, April 13–17, 2008, proceedings*, Lecture Notes in Computer Science, 4965, Springer, 2008. ISBN 978-3-540-78966-6. See [17].

[73] Douglas R. Stinson, *Some observations on the theory of cryptographic hash functions* (2001). URL: http://eprint.iacr.org/2001/020. Citations in this document: §B.6, §Q.20.

[74] Edlyn Teske, *On random walks for Pollard's rho method*, Mathematics of Computation **70** (2001), 809–825. URL: http://www.ams.org/journals/mcom/2001-70-234/S0025-5718-00-01213-8/S0025-5718-00-01213-8.pdf. Citations in this document: §3.1.

[75] Serge Vaudenay, Amr M. Youssef (editors), *Selected areas in cryptography: 8th annual international workshop, SAC 2001, Toronto, Ontario, Canada, August 16–17, 2001, revised papers*, Lecture Notes in Computer Science, 2259, Springer, 2001. ISBN 3–540–43066–0. MR 2004k:94066. See [50].

[76] Michael J. Wiener, *The full cost of cryptanalytic attacks*, Journal of Cryptology **17** (2004), 105–124. ISSN 0933–2790. URL: http://sites.google.com/site/michaeljameswiener/. Citations in this document: §Q.8.

[77] Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, Tal Malkin (editors), *9th international conference on theory and practice in public-key cryptography, New York, NY, USA, April 24–26, 2006, proceedings*, Lecture Notes in Computer Science, 3958, Springer, 2006. ISBN 978–3–540–33851–2. See [27].

A Appendix: Review of cost metrics for algorithms

Recall from Section 1 that concrete security definitions depend implicitly on how the "cost" of an algorithm is defined. This appendix reviews three cost metrics from the literature: the RAM metric; the NAND metric; and the *AT* metric. This is not meant to be a comprehensive survey of cost metrics in the literature; it does not include, for example, the volume-time metrics studied by Rosenberg in [**68**] and by Preparata in [**65**].

A.1. The RAM metric. Bellare, Kilian, and Rogaway in [**8**, Section 2.2] fix "some particular Random Access Machine (RAM)" as a model of computation. They define the running time of an algorithm A as "A's actual execution time plus the length of A's description".

There are well-known difficulties in giving a reasonable definition of "execution time" for RAM programs. However, standard workarounds (see, e.g., [**21**]) limit these difficulties to a much smaller scale than the gaps considered in this paper, so we do not review the details. We make an exception in Section 5, where the gap is relatively small.

Bellare, Kilian, and Rogaway say that adding the length of the algorithm "eliminates pathologies caused if one can embed arbitrarily large lookup tables in A's description". The obvious example is an algorithm that includes a giant sorted table of pairs $(\mathrm{AES}_k(0), k)$ for all 2^{128} AES keys k, and simply looks up $\mathrm{AES}_k(0)$ in the table to find k; the RAM metric forces this algorithm to pay for the length of the table, not merely the time taken for the table lookup.

It is interesting to note that [**7**], the original Crypto '94 version of [**8**], simply used execution time as a metric. Apparently it was not immediately obvious that the metric would allow exactly these "pathologies", posing huge problems for security definitions, theorems, and conjectures using that metric. The fix used in [**8**], and in many other papers, was to change the metric.

The more advanced attacks presented in Sections 2, 3, 4, and 5 can be viewed as similar "pathologies" that, contrary to the claim in [**8**], are *not* eliminated by merely adding the length of the algorithm. This view raises the question of whether further changes to the cost metric could stop those attacks.

A.2. The NAND metric. Bellare, Kilian, and Rogaway also consider an "alternative" definition of an algorithm as a circuit "over some fixed basis of gates, like 2-input NAND gates". The cost of an algorithm then "simply means the circuit size".

Counting NAND gates is refreshingly precise and easy to define. Readers might wonder why this NAND metric is an "alternative" rather than the standard definition of algorithm cost. The only answer given in [8] is that the NAND metric is "rather less intuitive" than the RAM metric.

We emphasize that the NAND metric often assigns far larger costs to algorithms than the RAM metric does. In some cases an algorithm taking time T in the RAM metric costs more than T^2 NAND gates. The most important difference is in the cost of random access to a large table, a very fast operation in the RAM metric but a very slow operation in the NAND metric. A batch of n independent random accesses to the same table of size n has similar cost in both metrics (since it can be simulated by sorting), but many algorithms require serial random accesses.

This difference can cause trouble: there are many theorems regarding "time" that are true for the RAM metric but unproven, and presumably false, for the NAND metric. This trouble is described in more detail in Appendix B.2. However, for the same reason, one can hope that any "pathologies" in the RAM metric are fixed by the NAND metric. This hope is analyzed in Sections 2, 3, 4, and 5.

A.3. The *AT* metric. In hardware design it is common to model computation in a completely different way. Computation is performed by a chip, i.e., a rectangular mesh of transistors connected by wires, with at most a few layers of wires at each point in the mesh. Transistors and wires all operate in parallel. It is not difficult to give a formal definition of this model of computation; see, e.g., [24]. This definition has the virtue of being obviously quite close to the physical reality of how computations are actually performed. See [19] for a detailed evaluation of many models of computation from this perspective.

Our third cost metric for algorithms is the price-performance ratio of a chip: i.e., the product AT of the area A of the chip and the time T taken by the chip. Hardware designers often consider more general functions of (A, T), but the classic product AT remains the default choice of cost metric in thousands of papers because it preserves the following two forms of linearity: performing n time-T computations in serial on one area-A chip costs n times as much as performing 1 computation; performing n time-T computations in parallel on n area-A chips (formally, one area-nA chip) also costs n times as much as performing 1 computation.

Notice that the energy used by a computation is proportional to AT if the entire chip is active. This energy use (and corresponding heat dissipation) does not pose a scalability problem as the chip area increases: building a corresponding area of solar cells and batteries and heat sinks increases the chip cost by at most a small constant factor. The energy use can be considerably smaller than AT if the chip is mostly inactive — if most of the transistors are, in the words of [14], "simply sitting around, twiddling their thumbs" — but any such chip is obviously highly suboptimal: for essentially the same investment in chip area one can build a much more active chip that stores the same data and that at the same time performs many other useful computations. This trend towards increasing parallelism can easily be observed in mass-market CPUs and GPUs: for example, a 2.27-billion-transistor Intel Xeon E5-4650 has 64 parallel 32-bit adders (8 independent cores, 2 independent vector units in each core, 4 synchronized 32-bit adders in each vector unit), and a 3.5-billion-transistor NVIDIA GeForce GTX 680 has 1536 parallel 32-bit adders.

Brent and Kung showed in [24] that n-bit multiplication costs $n^{1.5+o(1)}$ in the AT metric; for comparison, n-bit multiplication costs only $n^{1+o(1)}$ in the RAM metric and in the NAND metric. Similar comments apply to sorting and to various other high-communication computations. We consider the AT metric in Sections 2, 3, 4, and 5 for the same reason that we consider the NAND metric: it is a source of trouble but also a possible solution to "pathologies".

B Appendix: Trying to salvage the insecurity metric

As a followup to the algorithm analyses in Sections 2, 3, 4, and 5, this appendix analyzes the merits of five possible responses to the gap between standard-definition insecurity and actual insecurity. None of the responses are completely satisfactory, but some of them are arguably better than others.

B.1. Response 1: keep the definitions and abandon the conjectures. One possible response is to defend the metric, arguing that the attacks described in Sections 2, 3, 4, and 5 actually *should* be viewed as assigning security levels far below 2^{128} to AES, NIST P-256, DSA-3072, and RSA-3072. In other words, this response is that standard-definition insecurity *is* actual insecurity, and that taking the cost of precomputation into account would be understating actual insecurity.

This response has the virtue of minimizing the number of changes required to the literature. The other responses considered below require revisiting every proof to see whether the theorem can still be proven in a new metric and to see exactly what quantitative changes are required; this response preserves the metric. Of course, the security conjectures made in [**11**, Section 1.4], [**12**, Section 3.6], [**5**, Section 3.2], etc. would still have to be withdrawn.

There are, however, several obvious problems with this response. First, real-world attackers have no choice but to pay for precomputation time, contrary to the standard definition. Why should cryptographers be more concerned about a time-2^{60} attack that takes time 2^{300} to find than about a time-2^{70} attack requiring no effort to find? Users aiming for the best possible security, subject to performance constraints, should prefer a system where the best attack is of the first type over a system where the best attack is of the second type; underestimating attack cost by ignoring precomputation cost will lead those users to select the wrong system, hurting their own security.

As a concrete example, consider the result of Menezes, Teske, and Weng in [**57**] that many composite fields are "weak fields for ECC". For example, every curve over $\mathbf{F}_{2^{210}}$ has "a security level of at most 91 bits" rather than the expected "104 bits". Every ECC researcher will agree that it is much safer to use the field $\mathbf{F}_{2^{211}}$. However, the standard definition of insecurity paints a quite different picture. The standard definition says that $\mathbf{F}_{2^{210}}$ has security level only 2^{70}; it ignores the fact that *finding* this 2^{70} attack costs 2^{140}; it says that the explicit 2^{91} attack of [**57**] is useless, since $2^{91} > 2^{70}$; and it says that $\mathbf{F}_{2^{211}}$ offers only a tiny security benefit.

Furthermore, almost every cryptanalytic paper includes the cost of precomputation, contrary to the standard definition. The standard security conjectures would be perfectly reasonable if the cost of precomputation were included but are false according to the standard definition. In short, there is a perfect alignment between what the conjectures say, what the cryptanalysts are looking at, and what real-world attacks can actually do; what is out of whack with this picture is the standard definition.

The core argument for confidence in security conjectures about (e.g.) RSA-3072 is that RSA-3072 has survived extensive cryptanalysis. The standard definition makes this argument untenable. There has *not* been extensive study of attacks against RSA-3072 that exploit free precomputation. Our new RSA-3072 attack in Section 5.3 is considerably faster than Coppersmith's original attack, and we would not be surprised to see substantial further speedups; any conjecture in this model would be built upon quicksand.

B.1.1. The amortization-eliminates-precomputation fallacy. One might argue that precomputation should be ignored because it can be amortized across many targets. However, this argument confuses two different concepts. Non-uniform attacks and multiple-target attacks (and non-uniform multiple-target attacks) are often quantitatively and qualitatively different: non-uniform attacks often benefit from vastly larger precomputation (as illustrated by the doubly exponential cost $\exp(2^{2n+1})$ to find the attack D_s in Section 2.2), while multiple-target attacks often benefit from batching (as illustrated by Section 2.4).

As a concrete example, consider a time-2^{170} precomputation of a time-2^{85} attack (such as the ECC attack in Section 3), which is then applied to 2^{40} targets. The total attack cost is 2^{125} (i.e., 2^{85} per target), but this is completely unnoticeable compared to the precomputation cost of 2^{170} (i.e., 2^{130} per target). Highlighting the 2^{125} rather than the 2^{170} makes no sense. The picture does not change for 2^{60} or even 2^{80} targets. Note that it is easy to imagine real-world attack power growing to 2^{85} curve operations (for comparison, standard technology would carry out 2^{87} bit operations per year using the 65-megawatt power supply of NSA's reported new Utah data center), while it is far more difficult to imagine how the number of real-world targets could grow to 2^{80}.

B.1.2. The more-conservative-is-better fallacy. One might also argue that ignoring precomputation is "more conservative" than taking precomputation into account, and in general that underestimating the cost of an attack is perfectly safe, since it simply leads users to choose larger parameters.

In fact, "conservative" underestimates can cause users to *lose* security. There are three important effects ignored in the "more conservative is better" argument: first, users are subject to cost constraints, and cannot simply choose larger parameters; second, users *can* choose different systems, and in fact take advantage of this flexibility with the goal of maximizing security subject to the cost constraints; third, underestimates in general vary from one system to another, and in particular the gaps considered in this paper vary from one system to another.

The following example illustrates all of these effects. Consider a DSA user who can just barely afford $p \approx 2^{3072}$ and $q \approx 2^{256}$. Both p and q are important for speed: DSA cost is approximately linear in $\log q$ and worse than linear in $\log p$. These parameters are commonly recommended as providing about 2^{128} security, but this recommendation takes precomputation into account, in violation of the standard definition. The standard definition says that the q attack described in Section 3 reduces security to about 2^{85}. A user deceived by this underestimate will increase q to gain security against this attack, but is then forced by cost limitations to reduce p, and at some point the p attack described in Section 4 becomes more worrisome. A detailed analysis suggests that $p \approx 2^{2705}$ and $q \approx 2^{330}$ is optimal, balancing the standard-definition cost of these attacks at about 2^{110}. However, taking precomputation into account shows that the user has now *lost* several bits of security.

Improvements in either Section 3 or Section 4 would change all the details of this example. For example, moderate improvements in Section 4 might bring 2^{3072} and 2^{256} back into balance, eliminating the security loss, but further improvements in Section 4 would then mislead the user into *decreasing* q below 2^{256}, again losing security.

B.2. Response 2: switch to the NAND metric. Another possible response is to change the algorithm cost model from the RAM metric to the NAND metric.

This response would cause trouble for the literature on provable concrete security. (All of the responses below would also cause various levels of trouble.) Proofs would have to be reviewed for RAM-dependent cost analyses, and many theorems would have to change, because many reductions would become much more expensive. For example, eliminating repeated queries to an oracle is a very common step in security proofs; it is practically free in the RAM metric (add each query into a fast associative array) but much slower in the NAND metric. In other words, even though the NAND metric was mentioned as an "alternative" in [8], the literature did not develop in a way consistent with this alternative.

The motivation for this response, as mentioned in Appendix A, is the hope that this response would fix the "pathologies" in the RAM metric: i.e., that the gap between actual security and the standard definition of insecurity is an artifact of the low-cost random access provided by the RAM metric. However, the analyses in Sections 2, 3, 4, and 5 do not provide any support for this idea. All necessary random accesses appear in large batches, allowing reasonably efficient NAND computations.

B.3. Response 3: switch to the *AT* metric. Another possible response, analogous to the previous response but different in one critical detail, is to change algorithm cost model from the RAM metric to the AT metric.

Our cost analyses provide reason to hope that this response *does* fix essentially all of the pathologies in the RAM metric. There is an exception in one corner case — precomputation appears to help PRP attacks for probabilities below $2^{-K/4}U^{-1/2}$, where K is the key length and U is the number of targets — but one can argue that such low-probability attacks are of no concern for cryptographic users.

This response causes trouble for the literature on provable concrete security, similar to the previous response but even more pervasive: even more theorems would have to change. Like the NAND metric, the AT metric makes serial random access much more expensive than the RAM metric; unlike the NAND metric, the AT metric makes a large batch of table accesses much more expensive than the RAM metric.

B.4. Response 4: add constructivity. In provable security (and in complexity theory more broadly) it is standard to formalize "one can find a cost-C algorithm A that breaks X" as "there exists a cost-C algorithm A that breaks X". This formalization ignores the question of how *difficult* it is to find A. Another possible response is to switch to another formalization that explicitly quantifies this difficulty.

The obvious way to quantify the hardness of finding A is as the minimum cost required by all algorithms B that print A. The obvious objection is that there is always a cost-approximately-C algorithm to print A: namely, an algorithm that simply includes, and prints, a copy of A. However, one can easily exclude this trivial algorithm by allowing only *small* algorithms B.

Consider, for example, a large chip containing billions of standard AES key-search units. This chip breaks AES with AT cost roughly 2^{128}. The chip has a regular structure and area far below 2^{128}, so it is printed at moderate cost by an even smaller chip B. Similar comments apply to the standard chips to attack NIST P-256, DSA-3072, and RSA-3072 at cost roughly 2^{128}: all of them are printed at moderate cost by small chips B.

Consider, as another example, the hard-to-find attack A of Section 2.3, which finds an AES key with high probability using 2^{43} tables, each of size 2^{43}, for a total RAM cost on the scale of 2^{86}. The description of A in Section 2.3 is a small algorithm B that prints A, but B has RAM cost on the scale of 2^{128}. One can trade some space for time by embedding part of A into B, but as far as we know every algorithm B significantly smaller than A has negligible chance of printing A with RAM cost significantly below 2^{128}.

Consider, as a third example, the hard-to-find chip A of Section 2.2, with AT cost about 2^{3n}: area about 2^{2n} and time about 2^n. As far as we know, every significantly smaller chip B has negligible chance of printing A in any tolerable amount of time. This example suggests that it is possible to eliminate some corner pathologies that were not eliminated by merely switching to the AT metric.

Note that it is important to limit the cost of B in some reasonable cost metric, not merely the size of B. All of the precomputations considered in this paper can be carried out by rather small RAM algorithms; in other words, the outputs have low Kolmogorov complexity. For the same reason, the NAND metric is useless for measuring the cost of B.

In general, it seems reasonable to redefine the insecurity of X as the maximum, over all size-limited cost-limited algorithms B that print cost-limited algorithms A, of the probability that A succeeds in breaking X. This definition is explicitly parametrized by three numerical limits, and implicitly parametrized by the metrics used to specify those limits. We emphasize that probability here considers not just the randomness in X and in A, but also the randomness in B; otherwise the best choice of B is a tiny algorithm that prints out random bits. If B is required to be a *deterministic* algorithm then imposing a size limit and cost limit on B is, aside from polynomial factors, equivalent to imposing limits on A under some of the notions of time-bounded/resource-bounded Kolmogorov complexity surveyed in [3]; but polynomial factors are important, and excluding probabilistic algorithms does not seem safe for cryptography.

This response stops all of the precomputations considered in this paper. For example, it allows the feasible low-probability attack considered in Section 2.1 but does not allow the precomputed attacks considered in Section 2.2.

This response, like the previous two responses, causes trouble for the literature on provable concrete security. Each theorem must be restated to track not only the cost of A but also the size and cost of B.

B.4.1. Evaluating the realism of constructivity. Any limitation on the set of attacks considered by security definitions raises the question of whether real attackers are in fact bound by that limitation. Observing that today's state-of-the-art attacks obey the limitation does not resolve the question; perhaps a new attack tomorrow will violate the limitation, and will break a system that was declared to be secure by the limited definition.

One way to argue for the constructivity of real attack algorithms is to observe that real attacks are found by humans; except for minor implementation details, humans are simply chips, and rather small chips by cryptanalytic standards. Of course, one can imagine humans building larger chips that in turn find algorithms that the humans would not have found directly, and to model this one can consider longer chains such as algorithms that print larger algorithms that in turn print larger algorithms; but it seems reasonable to insist that the chain start with a small algorithm (small enough for humans to find) and to put a cost limit on each algorithm. One can also compress these chains after the second step, replacing a large algorithm that *prints* A by a large algorithm that *simulates* A, so there is only a small cost difference between (e.g.) the length-2-chain theory and the length-3-chain theory.

A counterargument is that many humans working together in fact form quite a large chip and have already developed impressively large algorithms, such as the multiple-gigabyte software collection shipped with today's operating systems. Even if all of today's state-of-the-art attack machines are printed by programs considerably smaller than a megabyte, humans are certainly not limited to building such programs.

A different, more fundamental, argument for constructivity is as follows. Consider a chip that simulates a randomly initialized world according to, e.g., Conway's Game of Life [36], or a better approximation of the laws of physics. One can reasonably conjecture that, if the simulated world is large enough, it will contain simulations of some simple life forms that randomly evolve into more complex life forms, eventually reaching the intelligence of the human race, even though this chip is quite simple and is easily printed by a very small algorithm. One can also modify the chip to communicate to the simulated life forms their job of attacking a particular cryptosystem (or carrying out some other algorithmic task), and to record their best results; this communication does not require much extra complication in the algorithm that prints the chip, beyond a description of the cryptosystem being attacked.

There are several ways that such simulators can fail: perhaps the simulated rules are not actually complex enough to impose evolutionary pressures that create intelligence, or perhaps the simulated world is too small to support the intelligence required to find the real attacker's best attack, or this evolution is too slow, or the final simulation of the attack is too slow. However, one can plausibly conjecture reasonably small bounds on each of these obstacles, and continued research into artificial intelligence can be expected to add evidence for such conjectures. We also comment that quantum computation allows analogous simulators and conjectures.

To summarize, the fact that today's state-of-the-art attack machines can be printed at reasonable cost by small algorithms is not merely an accidental feature of those attacks; it follows from the general evolutionary conjecture that an adequate simulation of the world can be printed by a small algorithm.

B.4.2. Formalizing collision resistance. We comment on a surprising consequence of this notion of constructivity: there is an apparently reasonable definition of collision resistance for constructive deterministic hash functions with large output sizes. Specifically, we define the collision insecurity of a hash function H as the maximum, over all size-limited cost-limited algorithms B that print cost-limited algorithms A, of the probability that A prints a collision in H.

The conventional wisdom is that collision resistance is unformalizable for any specific hash function H small enough to be computed. Specifically, if H has n-bit output, then there certainly exists a collision between two $(n+1)$-bit inputs, and therefore there exists an algorithm A that simply prints those two inputs.

However, this algorithm A contains more than $2n$ bits. One can easily reduce $2n$ to n, but if n is at least (say) 1024 plus the size limit on B then there is no obvious way for B to print A or any other fast collision-finding algorithm.

We do not claim that this definition is meaningful for hash output sizes as small as (e.g.) 512 bits; formalizing the collision resistance of SHA-512 remains an open problem. However, the definition does appear to be meaningful for hash output sizes larger than the sizes of the simulator-printing algorithms discussed above, say 1 megabyte.

One can try to separate this formalization from an intuitive notion of collision resistance as follows. Assume that a hash algorithm H is collision-resistant according to this definition; generate a "back door" consisting of two distinct gigabyte-long random strings r_1, r_2 and an n-bit random string s; define a new function H' that maps both r_1 and r_2 to s and that is otherwise identical to H. A collision in H', namely (r_1, r_2), is known to the manufacturer of H' and to anyone who looks at the most obvious program for H'; but clearly no small program B will print a program A that has a noticeable chance of finding this collision.

However, this function H' is itself not constructive: there is no size-limited cost-limited algorithm that prints a cost-limited algorithm that computes H'. One can try to model collision resistance for such functions by allowing A to take a hash-computing program as an extra input, or one can simply restrict attention to constructive hash functions, where such issues do not seem to arise.

B.5. Response 5: add uniformity. The final response we consider is to eliminate non-uniform security definitions: to prevent precomputation by requiring a single attack algorithm to work against many different cryptographic systems.

The classic form of uniformity considered in the computational-complexity literature is size-uniformity. One considers, for example, an attack against the entire RSA family (a single algorithm that takes as input an RSA key of any length), not just RSA-3072. One defines RSA to be (t, ϵ)-secure if every attack taking time at most t has success probability at most ϵ; here both ϵ and t are functions of the length of the RSA modulus.

Observe that this approach abandons the goal of defining, e.g., the insecurity of RSA-3072. Substituting 3072 into ϵ and t does not work: it allows exactly the same precomputations as in Section 5.

An alternative, already used in common definitions of collision resistance, is to consider uniformity across wider families of functions. There is no longer a definition of the security of AES; instead there is a definition of the security of a family of 2^{128} variants of AES. This security depends on the choice of family. One might try to define a family of functions "similar" to AES, hoping that the uniform security of this family reflects the actual security of AES; but cryptanalysts have little motivation to study the family, so building confidence is difficult. For RSA-3072 the situation is even worse: any reasonable family of 3072-bit functions arguably sharing the security of RSA-3072 seems to be vulnerable to the same precomputations as RSA-3072. For elliptic-curve cryptography the situation is somewhat better, since one can reasonably ask questions about the security of (e.g.) a random curve meeting the IEEE P1363 criteria over a randomly chosen 256-bit prime field; however, this is of no help in understanding the security of the NIST P-256 curve.

It is clear that insisting on enough uniformity, taking enough steps away from specific cryptographic primitives towards sufficiently diverse families of cryptographic primitives, would eliminate the gap analyzed in this paper. The gap relies on non-uniformity, and we have chosen to highlight non-uniformity in the title of this paper.

The fundamental problem with this response is that it disconnects provable security from cryptographic reality. For almost twenty years the literature on provable concrete security has promised to formally define and study the security of specific cryptographic systems of interest to cryptographic users and cryptanalysts, such as AES and AES-CBC-MAC and RSA-3072. Adding uniformity would abandon this promise. Without these definitions it is unclear how to make meaningful statements comparing the security of two different ciphers, or two different curves, or any two cryptographic protocols that are specific enough to actually be used in practice.

This fundamental problem is already well known for the special case of collision resistance. For example, Rogaway in [**67**, Section 2] comments that uniformity conditions render collision-resistance definitions inapplicable to specific hash functions such as SHA-256 and in particular "distance the definition from the elegantly simple goal of the cryptanalyst: publish a collision for the (one) function specified by NIST."

A second problem with this response is that it forces syntactic changes to most theorems. For example, instead of proving a theorem comparing the security of a block cipher F to the security of CBC^m-F as in [**8**], one would have to prove a theorem comparing the security of a family of block ciphers to the security of the corresponding CBC family.

B.6. Recommendations. We believe that accurately modeling reality is more important than minimizing the number of changes required to the literature. We recommend switching to the AT metric (response 3), capturing real limitations on communication cost that are ignored by the RAM metric and the NAND metric. We also recommend adding constructivity (response 4), capturing the fact that attackers are limited in precomputation cost. We recommend against adding uniformity (response 5); users are in fact using AES and NIST P-256 and RSA-3072, not large families of variants of AES and NIST P-256 and RSA-3072.

We recommend stating provable-security theorems in a way that minimizes the hassle of switching to a new cost metric. For example, consider again the main theorem of [**8**], comparing the security of a block cipher F to the security of CBC^m-F. To prove this theorem one compares $\mathrm{UseCBC}(A)$ to A in cost and in success probability, where A is any attack against CBC^m-F and UseCBC is a particular reduction producing an attack $\mathrm{UseCBC}(A)$ against F. The comparison of success probability is independent of the cost metric, and we recommend stating it as a separate theorem that can be reused for different cost metrics. The following theorem from [**17**] illustrates how simple such statements can be:

"**Theorem 3.1.** $\mathrm{PrFactor}(\mathrm{RandSquare}(A)) \geq (1/2)\,\mathrm{PrInvBlind}(A)$."

The reduction RandSquare is defined before the theorem, and PrFactor and PrInvBlind are two types of success probabilities. This theorem is independent of cost metric, and is easy to reuse in various higher-level theorems that compare the insecurity of the objects attacked by A and $\mathrm{RandSquare}(A)$ in various cost metrics: the proof of a higher-level theorem analyzes the relative costs of A and $\mathrm{RandSquare}(A)$ and appeals to this theorem for the relative success probabilities. Similarly, the main security theorem of [**8**] factors easily into (1) a lower-level theorem stating

Theorem. $\mathrm{AdvPRF}_{\mathrm{CBC}^m(F)}(A) \leq \mathrm{AdvPRP}_F(\mathrm{UseCBC}(A)) + q^2m^2/2^{l-1}$

and (2) a cost comparison of $\mathrm{UseCBC}(A)$ with A. Changing the cost metric (for example, to switch to the AT metric and add constructivity as we recommend, or to make further changes) then requires merely redoing the cost comparison and the main security theorem; the lower-level theorem is unaffected and can simply be reused.

This type of modularization of provable-security theorems can be traced back to at least 1999 (see [**13**, Theorem 4.1]), if not earlier, but at that point was not claimed to have any particular advantages. In 2001, Stinson proposed studying explicit hash-function reductions as a workaround for the separation between non-uniform collision resistance (which is negligible) and uniform collision resistance (which often appears to be very high); but Stinson's collision-resistance theorems (e.g., [**73**, Theorem 3.1]) do not actually make these reductions explicit and are trivialities as stated. Rogaway in [**67**] analyzed the same proposal much more carefully, and gave examples of nontrivial theorems about black-box and non-black-box reductions; but these theorems were still monolithic, handling probability together with a particular cost metric and encapsulating the reductions inside the proofs, so changing the cost metric is unnecessarily difficult. With the modular approach we recommend, reductions such as UseCBC are defined outside theorems, so that one theorem can analyze the UseCBC success probability while another theorem analyzes the UseCBC cost.

We emphasize that this modularization does *not* mean abandoning concrete security definitions parametrized by attack cost, and does *not* mean abandoning cost analyses of reductions. On the contrary: it is obviously desirable to have meaningful "X is secure" statements as conclusions of theorems, and this is impossible without cost analyses and cost-based security definitions. Our proposed modularization *factors* each concrete-security theorem into two parts: a probability theorem and a higher-level cost-vs.-probability theorem. The work inside the probability theorem is independent of cost metric; cost analysis is isolated inside the higher-level theorem.

Our analyses suggest that constructivity and the AT metric provide two levels of defense against the unrealistic attacks considered in this paper. However, we would not be surprised if further cost-model changes turn out to be desirable for other reasons, and we think that a modular style for provable-security theorems will reduce the effort required to make such changes in the future.

G Appendix: Graphs

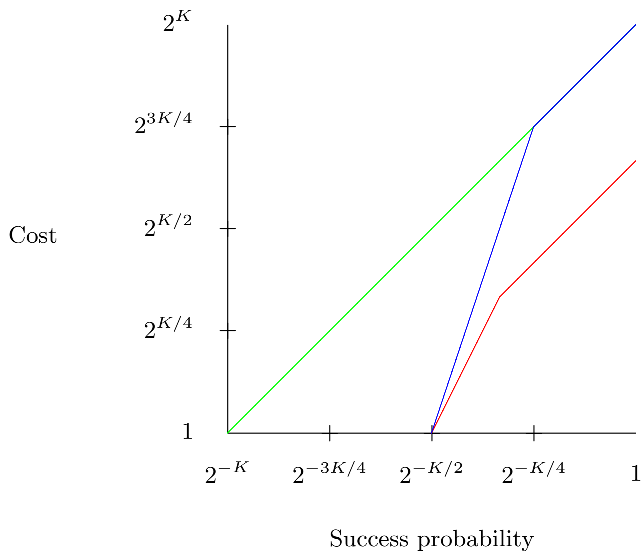

Fig. G.1. Cost summary of PRP attacks against one K-bit key. Horizontal axis: Attack success probability p, from $1/2^K$ to 1, on a logarithmic scale. Vertical axis: Attack cost, from 1 to 2^K, again on a logarithmic scale. Top curve (green): Cost $2^K p$, approximating cost of simple exhaustive search. Bottom curve (red): Cost $2^K p^2$ for $p \le 2^{-K/3}$ and $2^{2K/3}p$ for larger p, approximating RAM/NAND cost of best attack known. Middle curve (blue, merging with green): Cost $2^{3K/2}p^3$ for $p \le 2^{-K/4}$ and $2^K p$ for larger p, approximating AT cost of best attack known.

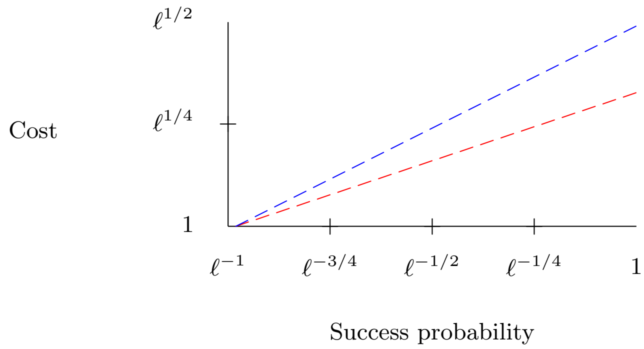

Fig. G.2. Cost summary of discrete-logarithm attacks for a group of size ℓ. Horizontal axis: Attack success probability p, from $1/\ell$ to 1, on a logarithmic scale. Vertical axis: Attack cost, from 1 to $\sqrt{\ell}$, again on a logarithmic scale. Bottom curve (red): Cost $(p\ell)^{1/3}$, approximating RAM/NAND cost of best attack known. Top curve (blue): Cost $(p\ell)^{1/2}$, approximating AT cost of best attack known.

L Appendix: Literature samples

> *The Basic Principles of Modern Cryptography ...*
>
> *Principle 1—Formulation of Exact Definitions*
>
> *One of the key intellectual contributions of modern cryptography has been the realization that formal definitions of security are* essential *prerequisites for the design, usage, or study of any cryptographic primitive or protocol.* —Katz and Lindell [45]

To collect data regarding standard practice in the literature on provable concrete security, we scanned the proceedings of Crypto 2012, searching for concrete security definitions and claims of provable concrete security. This appendix reports the results of this scan.

Most of the "provable security" papers do *not* prove anything about concrete security. Some use information-theoretic security notions that cannot be broken by any amount of attacker computation; the majority use purely asymptotic security notions such as "every probabilistic polynomial-time adversary has negligible success probability".

There are, however, five provable-concrete-security papers. Each of these five papers explicitly claims to "show" or "prove" or "guarantee" bounds that involve the "security" (or "insecurity" or "Adv") of some cryptographic systems. Proving such bounds obviously requires a quantitative definition of the "security" of a system; the central question here is what definitions the papers are actually using.

Inspection of these papers — see below for details — suggests that it is completely standard to use definitions that express exactly the following situation: there does not exist an algorithm that breaks the system with probability above ϵ while satisfying specified limits on "time", queries, etc. *None* of the five papers deviate from this standard. Three of the papers explicitly state new definitions that follow this standard. The other two papers use a classic special case, namely PRP "security", without defining it; after many years of papers repeating the same definition of PRP "security" — a settled definition that also follows this standard — the reader is forced to assume that these two papers are also using that definition.

L.1. Details. The five papers are as follows, in the same order that they appear on Springer's web site.

Jager, Kohlar, Schäge, and Schwenk in [43] give a "formal proof that [one version of TLS-DHE] is a secure authenticated key exchange protocol", define "the notion of authenticated and confidential channel establishment", and prove that a particular protocol "forms a secure ACCE protocol". [43, Definition 2] defines an encryption scheme to be (t, ϵ)-"secure" if "all adversaries A running in time at most t" have success probability at most ϵ; similar definitions continue throughout the paper. [43, Theorem 1] assumes that various primitives are (t, ϵ)-"secure" and concludes that “any adversary” that (t', ϵ')-breaks a particular protocol with “$t \approx t'$” must have $\epsilon' \le \cdots$. This is trivially equivalent to a conclusion that the protocol is (t', ϵ')-“secure” under the definitions in the paper.

Bellare, Ristenpart, and Tessaro in [9, Section 2] define a “(t, q, q_c)-adversary” as an algorithm that “runs in time t and makes at most $q[i]$ encryption queries of the form $\cdots$ and makes at most q_c corruption queries” and then define $\mathbf{Adv}^{\mathrm{uku}}_{\mathrm{SE},m}(t, q, q_c)$ as the maximum advantage among “all (t, q, q_c)-adversaries”. [9, Theorem 1] bounds “$\mathbf{Adv}^{\mathrm{uku}}_{\mathrm{SE},m}(t, q, q_c)$” relative to another similarly defined insecurity function.

Hoang, Morris, and Rogaway in [41, Section 4] briefly discuss the “complexity-theoretic interpretation” of an information-theoretic result. They do not state a formal theorem but they say that “the PRP-security of E is the PRF-security of F minus [something small]”. The paper neither gives nor cites a definition of “the PRP-security of E”, so (as above) the reader is forced to assume that the paper reuses the definition given in many previous papers, referring to a low success probability of all algorithms subject to specified limits on time and queries. Similar comments apply to [41, Section 5], which asserts that one is “guaranteed” that attacks have a success probability of “less than 10^{-10}” plus the “insecurity” of AES.

Landecker, Shrimpton, and Terashima in [51, page 16] say that “the bulk of the paper is dedicated to showing that” the proposed “CLRW2 TBC” is “a strong tweakable-PRP” under various hypotheses, one hypothesis being that the underlying cipher E “is a secure strong-PRP”. The paper neither gives nor cites a definition of “secure strong-PRP”, so (as above) the reader is again forced to assume the standard definition. This assumption is consistent with the formulation of the “main technical result” of the paper, [51, Theorem 1], which hypothesizes that A is “an adversary asking a total of q queries to its oracles, these of total length μ, and running in time t’, and concludes that “there exists an adversary B using the same resources” whose success probability has a particular lower bound in terms of the success probability of A. The *existence* of an algorithm with specified success probability, time, etc. is exactly what is logically required to prove insecurity under the standard definition.

Hofheinz and Jager in [42, Theorem 5] state, under various hypotheses, that a particular system is “(ϵ, t, μ, q)-IND-CCA secure”. This means, by [42, Definition 5], that “there exists no adversary” that (ϵ, t)-breaks the (μ, q)-IND-CCA security of the system, i.e., there exists no adversary “that runs in time t and wins with probability $1/2 + \epsilon$” using μ public keys and q queries.

L.2. Controversy. An earlier version of this paper, predating the scan described in this appendix, had already asserted that such definitions are completely standard. This assertion was based on what we had seen in many papers over many years, and we did not expect it to be controversial. In response we encountered a puzzling claim that such definitions are “not widespread” and that, as a result, most provable-concrete-security papers do not fall into the gaps that we have identified.

It was of course possible that our earlier selection of papers was biased by our own interests, but this new literature sample has no obvious bias and makes the “not widespread” claim quite difficult to believe. If such definitions are “not widespread” then why are they used in all five of the provable-concrete-security papers at a recent IACR flagship conference — papers from Bellare, Hoang, Hofheinz, Jager, Kohlar, Landecker, Morris, Ristenpart, Rogaway, Schäge, Schwenk, Shrimpton, Terashima, and Tessaro? It is of course *still* possible that these five papers are not representative of standard practice in the literature as a whole, but we think that anyone making such a claim is required to supply a considerable volume of evidence.

Q Appendix: Frequently asked questions

This appendix answers several questions that we have been asked regarding this paper.

Q.1. In the real world, an attack is applied to many targets. Doesn't this make the precomputation effectively free? No, not in general. The fallacy here is addressed in Appendix B.1.1.

Q.2. Aren't we simply making the user safer if we underestimate attack cost by ignoring precomputation? No. The fallacy here is addressed in Appendix B.1.2.

Q.3. Why are you analyzing the cost of precomputation in these attacks? Didn't you just say that this cost is irrelevant to the security definitions? The security definitions are flawed. Seeing the cost of precomputation (e.g., 2^{128} for the AES attack and 2^{170} for the NIST P-256 attack) is critical for understanding the real-world inapplicability of the attacks. We recommend fixing the security definitions to take this cost into account; see Appendices B.4 and B.6.

Q.4. I don't understand how MD5 could possibly break AES. Are you sure? Yes. The analysis is easy, using standard heuristics, and is backed by the computer experiments reported in Appendix V.

Q.5. If one special precomputation is enough to completely break AES, isn't that serious enough that the metric should capture it? Of course. Success probability is important, and the number of targets is important. But the difficulty of carrying out the precomputation is also important, and is ignored in the standard security definitions. We propose taking all of these features into account. See Appendix B.6.

Q.6. Sorting costs $n \log_2 n$. How can you take a metric seriously if it says that sorting costs $n^{1.5}$? The best reported results for the energy consumption of sorting [58] are 43700 items sorted per joule for 10^{10} 100-byte items, and 9700 items sorted per joule for 10^{12} 100-byte items. The ratio 43700/9700 is 4.5, and the chip evolution mentioned in Appendix A.3 can be expected to improve the small sort by a larger factor than the large sort, pushing the ratio up towards 10. For comparison, the ratio $(\log_2 10^{12})/(\log_2 10^{10})$ is only 1.2, obviously understating the heavy communication costs of sorting. See [19] for a more detailed analysis of physical constraints upon algorithm performance.

Q.7. I spend time A setting up my area-A hardware, plus time T waiting for the results. Isn't $A + T$ my actual cost? If you have n processing cores, each carrying out a series of n separate computations, then you have carried out n^2 computations and used energy proportional to n^2, but $A + T$ is only linear in n. The RAM metric avoids this pathology by imposing an unrealistic prohibition upon parallel computations, but it falls prey to other pathologies.

Q.8. Isn't your AT metric simply a reinvention of Wiener's full-cost metric? It's not *our* AT metric. The AT metric has decades of history in algorithm analysis, and more than a decade of history in cryptanalytic algorithm analysis; see, e.g., [24] and [14]. Wiener's 2004 full-cost metric [76] appears to be equivalent to the VT metric analyzed in the 1983 papers [68] and [65]. See Q.9 for further analysis of VT.

Q.9. The world is three-dimensional! Isn't VT a more realistic metric than AT? The world is three-dimensional, but power consumption and heat dissipation are two-dimensional. A modern billion-transistor CPU is much more like a 32768×32768 square than a $1024 \times 1024 \times 1024$ cube; "three-dimensional" manufacturing technology is limited to a small number of layers (e.g., a flat $4 \times 16384 \times 16384$ box). Similarly, a "three-dimensional" computer cluster actually provides orders of magnitude faster communication within two-dimensional chips than it does between chips.

At a larger scale, each square meter of the Earth's atmosphere receives at most 1361 watts from the Sun, so one cannot hope to sustain an A-square-meter computer that consumes more than $1361A$ watts, no matter what the computer technology is. A three-dimensional VT-optimized computer does not have enough surface area to receive and dissipate the energy it uses, and if one tries to "spread it out" to give room for more energy then one ends up with VT cost matching AT cost. One might argue that it is possible to temporarily violate the $1361A$ maximum by shifting energy through time, as illustrated by oil drilling, or similarly by shifting energy around the Earth's surface, but this argument ignores the energy cost of the shifting mechanisms.

Furthermore, three-dimensional designs pose severe *temperature* problems even if enough energy magically becomes available at the surface of the computer. Increasing the size of a VT-optimized design, while still providing the energy needed for each bit operation, requires a corresponding increase in the amount of power that passes from the surface *through* each point of the design. This is incompatible with the temperature limits that are needed for adequate error correction in any physical system for computation. This is true even for limited three-dimensional systems that use only one layer for transistors and the rest for wires: wires require energy proportional to their length.

We have checked that the AT analyses of high-probability attacks in this paper would produce the same results if AT were replaced by VT, so the criterion of avoiding these "pathologies" is equally served by AT and by VT, but AT is much more accurate in modeling physical reality.

Q.10. Doesn't the physical reality actually say that each computation is $O(1)$? Quite possibly, yes. We recommend Aaronson's exposition in [**2**] of a physics argument concluding that the "maximum number of bits that could ever be used in a computation in the physical world" is about 10^{122}, inversely proportional to the cosmological constant. The two major steps in the argument are as follows. First, trying to pack information too densely would violate the Bekenstein–Hawking bound (see, e.g., [**4**]), which states that the entropy inside a spherical region of space is at most 1/4 of the surface area (not volume) of the region measured in Planck units. Second, as observed by Bousso in [**23**], information located too far away is forced by the nonzero cosmological constant to accelerate away so quickly as to become unobservable.

This bound on the number of bits used by a computation implies that every terminating physical computation has overwhelming probability of terminating within a particular constant number of bit operations. Standard asymptotic complexity classes such as P, BPP, BQP, etc. have no logical connection to such short computations. Aaronson's only suggestions for allowing a physical realization of the standard complexity classes are to switch to an imaginary alternate universe in which the cosmological constant is zero, or a sequence of imaginary alternate universes with cosmological constant converging to zero.

Fortunately, these asymptotic issues also have no logical connection to concrete security questions such as the security of AES-128, NIST P-256, DSA-3072, and RSA-3072. This paper focuses almost exclusively on concrete security questions, with asymptotics appearing only occasionally as inspiration. For example, Section 5.2 briefly analyzes the asymptotics of one algorithm, but this is merely a warmup for the concrete analysis of the same algorithm in Section 5.3.

Q.11. What... is the air-speed velocity of an unladen swallow? What do you mean? An African or European swallow?

Q.12. Have you considered effectivity? Yes. Effectivity is another name for constructivity, which is analyzed in Appendix B.4. We recommend a specific strategy (which appears to be new) for incorporating a quantitative form of constructivity into security definitions.

Q.13. When you say that the algorithms are non-uniform, aren't you really trying to say that the algorithms are non-constructive? Almost all of the algorithms we present are non-uniform: for example, the RSA attack requires different precomputations for sufficiently different modulus sizes, and the ECC attack requires different precomputations for different curves. Almost all of the algorithms we present are *also* non-constructive, i.e., very hard to find; see Appendix B.4 for a quantitative formalization of this intuition. Non-uniformity and non-constructivity are not the same concept.

Q.14. Isn't "non-uniformity" a purely asymptotic concept? No. Consider, for example, a theorem proving that for each 512-bit string s there is a fast algorithm A that, given a 512-bit string t as input, prints collisions in the 512-bit-to-128-bit function $x \mapsto \text{MD5}(s, t, x)$. Consider a stronger theorem proving that there is a fast algorithm A that, given 512-bit strings s and t as input, prints collisions in the 512-bit-to-128-bit function $x \mapsto \text{MD5}(s, t, x)$. The difference between these theorems is precisely in the amount of uniformity imposed on A. There are no asymptotics here.

Q.15. My favorite textbook defines "non-uniform" algorithms specifically as families of polynomial-size circuits, one circuit for each input size, and defines "uniform" algorithms specifically as polynomial-time algorithms. Aren't these definitions completely standard? Yes, these are two quite standard uses of the word "uniform", and often the only uses described in introductory textbooks, the same way that many introductory mathematics textbooks define "small" as comparing real numbers in the usual ordering. However, the complexity literature actually uses the name "uniform" for a much wider range of definitions, the same way that the mathematics literature uses "small" for a much wider range of definitions.

Example 1: Consider Goldreich's discussion in [**37**] of the "amount of non-uniformity" in a circuit family. This is a quantitative metric (the length of an advice string), easily capturing the introductory notions of "uniform" (advice limited to 0) and "non-uniform" (no limit on the advice) but also allowing analysis of intermediate amounts of uniformity.

Example 2: Consider Katz's discussion in [**46**] of "logspace-uniform" families of circuits. Compared to the introductory polytime-uniform concept, logspace-uniform restricts non-uniformity in another dimension, different from the length of an advice string.

Example 3: Consider Shor's discussion in [**71**, Section 2] of the "additional uniformity condition" used to define the complexity class BQP: a condition on families of real numbers used as constants in algorithms, yet another dimension of uniformity.

Example 4: Consider Koiran's comment in [**49**, Section 6] that the "only difference between VPSPACE^0 and uniform VPSPACE^0 is the nonuniformity of the coefficient function; VPSPACE is even more nonuniform since arbitrary constants are allowed".

Q.16. ? 42.

Q.17. Isn't the hashing attack in Section 2 outperformed by the linear-cryptanalysis approach of De, Trevisan, and Tulsiani? No. The speeds are identical at the level of detail of our analysis. The linear-cryptanalysis approach is cited in Section 2 for the advantage of provability, but the hashing approach is amply tested and has the advantage of simplicity.

Q.18. Isn't it already well known that there's a gap between attack algorithms that exist and attack algorithms that can be constructed? For one important problem, the problem of finding hash-function collisions, it is indeed very well known that "the best algorithm that exists" is not a reasonable model of "the best algorithm that can be found". For example, there is a fast algorithm that outputs collisions in SHA-512, but actually finding such an algorithm seems hopeless. (See Appendix B.4.2 for further analysis of this gap.)

However, hash-function collisions seem to be viewed as an exceptional case. The same model is widely viewed as reasonable for AES security, RSA security, etc., as illustrated by the conjectures from [**11**, Section 1.4], [**12**, Section 3.6], [**5**, Section 3.2], etc. Rogaway begins [**67**] by describing "The Foundations-of-Hashing Dilemma" with no indication that the dilemma extends beyond hashing.

Q.19. These AES-128 and NIST P-256 and DSA-3072 and RSA-3072 issues are just quantitative, right? Yes, they're "just" quantitative, but getting the quantitative details right has always been a major theme of the concrete-security literature. This paper is aiming at large gaps that affect wide portions of the literature; for comparison, the gaps in many well-known "tightness" papers are quantitatively smaller and apply to much narrower portions of the literature.

Bellare, Desai, Jokipii, and Rogaway wrote in [6] that when reductions are loose "one must use a larger security parameter to be safe, reducing efficiency. Thus, in the end, one pays for inefficient reductions in either assurance or running time." Our attacks against AES-128, NIST P-256, et al. show that there is an even larger loss of "either assurance or running time" from a flaw in the standard security definitions: specifically, an inaccurate model of the set of algorithms available to the attacker.

Q.20. Doesn't Rogaway's "human ignorance" model already solve these definitional problems? No. Stinson in [73] proposed studying explicit hash-function reductions as a workaround for the difficulties of defining collision resistance, and Rogaway in [67] gave examples of nontrivial theorems following this proposal; in Appendix B.6 we review these proposals and recommend further modularization of such theorems. However, this line of work does nothing to solve the problem of finding a realistic formalization of the statement "SHA-512 is collision-resistant" — a problem that remains unsolved today (but see Appendix B.4). Similarly, this line of work does nothing to solve the problem of finding a realistic formalization of statements such as "NIST P-256 is secure" — a problem that was incorrectly believed to be solved many years ago and that is shown in this paper to be much more subtle.

Q.21. Are you saying that the *AT* metric fixes everything? No. Appendix B.3 evaluates switching to the *AT* metric; our analyses suggest that this switch fixes essentially all pathologies *except* in one corner case. Appendix B.4 evaluates adding constructivity; our analyses suggest that this fixes some corner pathologies. Appendix B.6 recommends both of these changes, and also recommends modularizing theorems to accommodate further changes.

Q.22. Are you seriously suggesting redoing all the security definitions and proofs in the literature to use constructivity and the *AT* metric? Yes. A few theorems are already factored in the way we recommend in Appendix B.6, but the vast majority of definitions and theorems require new work. Some proofs, such as the "coin-fixing" proofs criticized in [47], will not survive the transition to constructivity. Our impression is that most proofs in the literature are constructive, but tightness will often change dramatically (see, e.g., the comments on repeated queries in Appendix B.2), imposing surprisingly small limits on the number of queries that can be tolerated before a proof becomes vacuous and raising the question of whether there are tighter proof strategies. If a proof *has* been checked then the best way to record this fact is as a new theorem, so that users know what security guarantees they are actually being provided.

Q.23. Can't we just assume that *AT* theorems will look the same as current theorems? No. See Q.22.

Q.24. Can't we just assume that constructive theorems will look the same as current theorems? No, not always. See Q.22.

Q.25. How do you expect authors to learn how to do *AT* algorithm analyses? The *AT* metric is used in thousands of papers. We recommend the classic paper [24] for definitions and for illustrative algorithm-analysis examples.

Q.26. Isn't it inappropriate to switch definitions and start writing papers using the new definitions? There are ample precedents for this. For example, Newtonian physics was replaced when it was discovered to be a poor model of reality. We briefly review a much closer example from mathematics.

In the 19th century, Kronecker questioned the significance of proofs of existence that are not *effective*, i.e., proofs that do not explain how to find the object that allegedly exists.

The classic example is the Bolzano–Weierstrass theorem, which states that an infinite sequence of real numbers $x_0, x_1, \ldots \in [0, 1]$ has an infinite subsequence that converges. The critical observation in the usual proof is that there must be infinitely many i with $x_i \in [0, 0.5]$, or infinitely many i with $x_i \in [0.5, 1]$. Define I_1 as $[0, 0.5]$ if there are infinitely many i with $x_i \in [0, 0.5]$, otherwise as $[0.5, 1]$; define I_2 similarly as the left or right half of I_1; etc.; and, finally, take the infinite subsequence indexed by the first i with $x_i \in I_1$, the first subsequent i with $x_i \in I_2$, etc. Kronecker objected that this proof gives no way to *find* the desired subsequence: even if each x_i is completely explicit, there is no procedure to decide whether there are infinitely many i with $x_i \in [0, 0.5]$.

Early 20th-century formalizations of mathematics did not provide any way to express Kronecker's objection. The only formalization of "one can find x such that $p(x)$" was "it is not true that, for all x, not $p(x)$"; for many years it was not obvious that any other formalization was possible. However, the introduction of "constructive mathematics" showed that one *can* formalize mathematics in a way that gives different meanings to these two notions, disallowing the Bolzano–Weierstrass proof (and theorem) while preserving more explicit mathematical proofs. This is part of the background for our quantitative approach to constructivity.

Q.27. I reviewed a previous version of this paper and told you X. Why didn't you change your paper? Chances are excellent that we did add something to our paper in response to your review, either to comment positively on X or to explain why X is incorrect. In many cases we've added FAQ entries to help you find the answer. If you still haven't found the answer, please look for comments on your review in the following web page: http://hyperelliptic.org/tanja/nonuniform-reviews.txt

V Appendix: Verification that MD5 breaks AES

For each 8-bit string k and each 128-bit string p define $E_k(p) = \text{AES}_{k,0}(p)$, where "$k, 0$" means k zero-padded to 128 bits. This cipher E is a scaled-down version of AES. The attacker's goal is to distinguish E from a perfect cipher $p \mapsto F(p)$, where F is a uniform random permutation of the set of 128-bit strings.

Consider the following very fast attack, a scaled-down version of the simple attack described in Section 2.1: take 16 bits of cipher output, zero-pad to 128 bits, feed the result through MD5, and take the first bit of the result (specifically, the bottom bit of the first byte of the result). Note that for this attack there is no difference between a uniform random permutation F and a uniform random function F.

This attack outputs 1 for exactly 32949 out of all 65536 16-bit strings: i.e., the average attack output against a uniform random permutation is $32949/65536 \approx 0.502762$. If this attack is applied to the first 16 bits of $E_k(0)$ then it outputs 1 for exactly 114 out of the 256 keys k; i.e., the average attack output against E is $114/256 \approx 0.445312$. The success probability of the attack against E is, by definition, the absolute value of the difference of these two averages, namely $3765/65536 \approx 0.057449$.

Table V.1 displays the results of similar experiments for K-bit ciphers for a range of values of $K \leq 64$. In each case $2K$ bits of cipher output are zero-padded to 128 bits and fed through MD5. The table is consistent with the theory that the success probability of the attack drops as roughly $1/\sqrt{2^K}$, and very far from consistent with the naive theory that all very fast attacks have success probability dropping as $1/2^K$.

The analysis in Section 2.1 states that (if K-bit keys do not produce surprisingly frequent collisions in $2K$ bits of cipher output) replacing MD5 with a uniform random function would have probability approximately $1 - \text{erf}(x\sqrt{2})$ of producing an attack whose success probability is at least $x/\sqrt{2^K}$; e.g., probability approximately 0.31731 of producing an attack whose success probability is at least $0.5/\sqrt{2^K}$, and probability approximately 0.98404 of producing an attack whose success probability is at least $0.01/\sqrt{2^K}$. With this replacement, the second average described above (the average attack output against E) is the average of 2^K independent uniform random bits and thus has a bell-curve distribution of width roughly $1/\sqrt{2^K}$, while the first average (the average attack output against a uniform random permutation) is the average of 2^{2K} independent uniform random bits and thus has a much narrower bell-curve distribution of width roughly $1/2^K$, so the difference will almost always be on the scale of $1/\sqrt{2^K}$.

K	uniform		cipher		success	scaled
1	1	0.250000	0	0.000000	0.250000	0.353553
2	7	0.437500	4	1.000000	0.562500	1.125000
3	27	0.421875	4	0.500000	0.078125	0.220971
4	117	0.457031	6	0.375000	0.082031	0.328125
5	491	0.479492	21	0.656250	0.176758	0.999893
6	2038	0.497559	37	0.578125	0.080566	0.644531
7	8195	0.500183	64	0.500000	0.000183	0.002072
8	32949	0.502762	114	0.445312	0.057449	0.919189
9	131279	0.500790	245	0.478516	0.022274	0.504003
10	524921	0.500604	498	0.486328	0.014276	0.456818
11	2098847	0.500404	1029	0.502441	0.002037	0.092197
12	8389411	0.500048	2020	0.493164	0.006884	0.440563
13	33555992	0.500023	4063	0.495972	0.004052	0.366706
14	134215688	0.499992	8070	0.492554	0.007439	0.952152
15	536855969	0.499986	16399	0.500458	0.000472	0.085383
16	2147481189	0.499999	32644	0.498108	0.001892	0.484228
17	?	0.500000?	65347	0.498558	0.001442?	0.522044?
18	?	0.500000?	131291	0.500835	0.000835?	0.427734?
19	?	0.500000?	261618	0.498997	0.001003?	0.726442?
20	?	0.500000?	523667	0.499408	0.000592?	0.606445?
21	?	0.500000?	1048417	0.499924	0.000076?	0.109795?
22	?	0.500000?	2097213	0.500015	0.000015?	0.029785?
23	?	0.500000?	4193272	0.499877	0.000123?	0.356316?
24	?	0.500000?	8386407	0.499869	0.000131?	0.537354?
25	?	0.500000?	16776146	0.499968	0.000032?	0.184718?
26	?	0.500000?	33552224	0.499967	0.000033?	0.269531?
27	?	0.500000?	67108562	0.499998	0.000002?	0.026068?
28	?	0.500000?	134239208	0.500080	0.000080?	1.311035?
29	?	0.500000?	268434302	0.499998	0.000002?	0.049805?
30	?	0.500000?	536878982	0.500008	0.000008?	0.246277?
31	?	0.500000?	1073756269	0.500007	0.000007?	0.311711?
32	?	0.500000?	2147489600	0.500001	0.000001?	0.090820?

Table V.1. Success probability of MD5 as an attack against zero-padded K-bit AES keys. "Uniform" is the number of $2K$-bit strings s such that $\text{bits}_1 \text{MD5}(s, 0) = 1$, where $s, 0$ means s zero-padded to 128 bits and bits_1 means the first bit. The subsequent column is "uniform" divided by 2^{2K}. "Cipher" is the number of K-bit strings k such that $\text{bits}_1 \text{MD5}(\text{bits}_{2K} \text{AES}_{k,0}(0), 0) = 1$, where bits_{2K} means the first $2K$ bits. The subsequent column is "cipher" divided by 2^K. "Success" is the success probability of the attack: the absolute difference between "uniform" divided by 2^{2K} and "cipher" divided by 2^K. "Scaled" is "success" times $\sqrt{2^K}$. "?" means that "uniform" was not computed but was estimated to be $2^{2K}/2$.

This analysis is consistent with the experimental results. The analysis also covers, e.g., truncating $E_k(0), E_k(1)$ to $2K$ bits and zero-padding to 256 bits for $64 \leq K \leq 128$. in this case there is a slight difference between a uniform random function and a uniform random permutation, but only on the scale of $1/2^K$.