Encrypting Problem Instances

Or ... , Can You Take Advantage of Someone Without Having to Trust Him?

*Joan Feigenbaum**

Computer Science Department
Stanford University
Stanford, CA 94305

1. Introduction

This paper describes ongoing work on the task of *encrypting problem instances*, also known as *computing with encrypted data*. A *problem* is specified by a function f and an *instance* by a value x in the domain of f. The scenario involves two people, A and B. A has instances $\{x_i\}$ of f to which she needs answers, but she lacks the resources to compute them. We use the term resources completely generally–she may be lacking time, space, algorithmic knowledge, or appropriate hardware, or she may simply be too lazy to implement a solution that she knows others have already implemented. B has the resources to compute $f(x)$ and is willing to let A use them, i.e., he is willing to send her $f(x)$ if she sends him x. She would like to take advantage of his generosity without having to trust him, i.e., she does not want to reveal any more about her data than she must in order to enable him to compute the correct answer. Intuitively, we say that f *is encryptable* if A can easily transform instance x into instance x', obtain $f(x')$ from B, and easily compute $f(x)$ from $f(x')$ in such a way that B cannot infer x from x'.

* The author did some of this work while at AT&T Bell Laboratories for the summer. During the academic year, she is funded by a Xerox Corporation Fellowship and a grant from the AT&T Bell Laboratories Graduate Research Program for Women.

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 477-488, 1986.

In the commutative diagram of Figure 1, the horizontal arrows $x \longrightarrow x'$ and $f(x') \longrightarrow f(x)$ represent computations done by A, and the vertical arrows $x \longrightarrow f(x)$ and $x' \longrightarrow f(x')$ computations that only B can do. B actually does the computation $x' \longrightarrow f(x')$ but not the computation $x \longrightarrow f(x)$ because A does not want him to know x. The instance x is called the *cleartext instance* and the instance x' the *encrypted instance.*

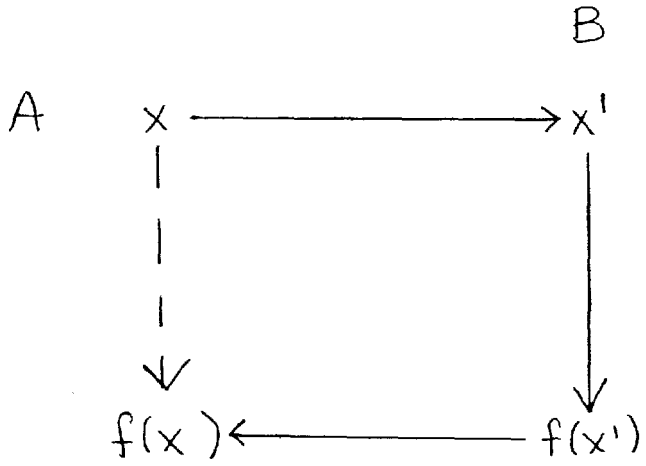

Figure 1. Because the diagram commutes, A learns the value of $f(x)$. A does the inexpensive computations $x \longrightarrow x'$ and $f(x') \longrightarrow f(x)$. B does the expensive computation $x' \longrightarrow f(x')$.

What follows is an attempt to formalize the statement "B cannot infer x from x'" and to give important examples of encryptable problems. Under one plausible definition of encryptability, all NP-complete problems that are P-isomorphic to CNF-SAT are encryptable.

It is important to keep the following aspects of the problem clear:

1) A's mistrust of B is confined to her fear that he will do something objectionable with her data; that is, she *does* trust him to give her the right answer $f(x')$ to her encrypted instance.

2) Encryptability is a property of the problem f, not of a particular solution to it. Thus, in her search for an encryption scheme, A cannot make assumptions about B's algorithm for computing $f(x')$.

3) A is not searching for a public key encryption algorithm. The security of x may rest on the fact that probabilistic choices she makes during the computation $x \longrightarrow x'$ are made in secret.

2. Motivating Examples

Example 1 is a scheme for encrypting instances of the discrete logarithm problem. As we go on, we will only consider formal definitions of encryptability that accept the scheme in Example 1, because it very clearly meets A's needs.

Example 1: Fix a large prime p and a generator g for the cyclic group Z_p^*. For $x \in Z_p^*$, A wishes to find the unique exponent $e = f(x)$ in $\{1, \ldots, p-1\}$ such that $g^e \equiv x \bmod p$. To encrypt the instance x, she chooses a random element c of $\{1, \ldots, p-1\}$ and lets

$$x' \leftarrow x \cdot g^c \bmod p.$$

She sends x' off to B, and decrypts the answer by computing

$$f(x) \leftarrow f(x') - c \bmod p - 1.$$

This scheme is obviously *feasible*: Simple arithmetic shows that she will get the correct value for the discrete logarithm of x, and both computations $x \cdot g^c \bmod p$ and $f(x') - c \bmod p - 1$ can be done in time polynomial in $\log p$. More importantly, it is also *secure*, because, for any pair of values x and x' in $\{1, \ldots, p-1\}$, there is an exponent c such that $x' \equiv x \cdot g^c \bmod p$. Thus we can say quite literally that, without knowing c, B cannot infer x from x'.

Example 2 is a scheme for encrypting instances of the clique problem that I'd like to reject because it's not secure. The analysis of Example 2 will point us to a precise definition of security. The scheme uses the following definition of graph multiplication: If G and H are undirected graphs without self-loops, then $G[H]$ ("G composed with H") is a finite graph with $V(G[H]) = V(G) \times V(H)$ and $E(G[H]) = \{(x,y) \text{ — } (v,w)\colon x \text{ — } v \in E(G) \text{ or } x = v \text{ and } y \text{ — } w \in E(H)\}$. By the *clique number* of a graph G, we mean the number k such that G has a clique of size k but no clique of size $k+1$.

Example 2: A's instance x is a pair G, k, where G is a finite, undirected graph without self-loops and k is an integer between 1 and $|V(G)|$, about which she inquires "does G have a complete subgraph with k nodes"? To encrypt x, A chooses a random graph H with clique-number j and computes

$$x' \leftarrow G[H],\ kj.$$

The answer to x' that she gets from B is the same as the answer to x, as the following lemma shows. Thus, the decryption step $f(x') \longrightarrow f(x)$ is trivial in this scheme.

Lemma 1: If H has clique number j, then G has a clique of size k if and only if $G[H]$ has a clique of size kj.

Proof: If $\{v_1, \ldots, v_k\}$ is a clique of G and $\{w_1, \ldots w_j\}$ is a clique of H, then $\{(v_a, w_b), 1 \leq a \leq k, 1 \leq b \leq j\}$ is a clique of $G[H]$. Conversely, if $C = \{x_1, \ldots x_{kj}\}$ is a clique of $G[H]$, then no more than j nodes in C can lie in a single copy of H. So C intersects at least k copies of H. Between any pair of these copies, there is at least one edge, and so all possible edges are present. Hence these copies correspond to at least k nodes of G that form a clique. ∎

The scheme in Example 2 is also feasible: A can grow H from one or more cliques of size j by adding only nodes of degree less than j, and she can construct $E(G[H])$ straightforwardly in time $O(|E(G)| \cdot |E(H)|)$. But we claim that the scheme is not secure. It is insecure because of the small number of possible cleartext instances that can correspond to an encrypted instance G', k'. Coppersmith and Feigenbaum show in [CF] that most composite graphs G' can be written as $G[H]$ for only one pair of graphs G, H and that if a graph has two inequivalent factorizations $G_1[H_1] \cong G_2[H_2]$, then $|V(G_1)| > |V(G_2)|$. Thus the number of possible cleartext instances that can be encrypted as G', k' can be very generously upper-bounded by the number of integer factorizations of $|V(G')|$ times the number of integer factorizations of k', which is all polynomial in $|V(G')|$. In the rare cases in which B cannot infer a unique G for which $G[H] \cong G'$, he can at least infer that G is a member of a small set.

But *how* can he infer this information? In other words, what is the complexity of factoring graphs under this definition of multiplication? Feigenbaum and Schäffer have shown that it is the same, to within polynomial factors, as the complexity of testing whether two connected graphs are isomorphic [FS].

Because there is no known polynomial-time algorithm for testing graph isomorphism, one is tempted to say that graph multiplication is a one-way function and hence this scheme is secure. Recall, however, that B is solving instances of the clique problem. So, unless P = NP, he *has* more than polynomial time and could decide to spend it decrypting x' rather than solving it.

In this crucial way, our version of computing with encrypted data departs from recent work on cryptography by the theoretical computer science community. We want to say what it means to encrypt instances of *hard* problems, for which B has to be given a lot of time, and hence cannot allow schemes whose security rests on intractibility *assumptions*. Rather than saying, as has been the fashion in computer science, that the cryptanalyst cannot decrypt the instance he sees because he does not have enough *time*, we want to return to more conventional criteria in cryptography and say that he cannot decrypt it because he does not have enough *information*. This is the case in Example 1, where B cannot figure out anything interesting because he does not know which value of c was used in computing x'.

3. A Precise but Lenient Definition of Encryptability

In this section, we explore the consequences of the lesson of Example 2. The graph-composition scheme fails because the number of cleartext instances that correspond to a given encrypted instance is too small. In the following definition of a successful encryption scheme, this situation is precluded explicitly.

Suppose for now that f is a decision problem. In her encryption algorithm E, A will combine elements of $Dom(f)$ with *keys* drawn from some convenient set K. The nature of K depends on the problem f. In Example 1, K was the set of exponents $\{1, \ldots, p-1\}$.

Definition 1: $E : Dom(f) \times K \rightarrow Dom(f)$ is a successful encryption function for the decision problem f if:

1) $E(x,k)$, $x \in Dom(f)$, $k \in K$, can be computed in time polynomial in $|x|$,

2) $E(x,k)$ is a yes-instance of f if and only if x is a yes-instance of f,

3) If x' is in the range of E, then

$$|\{x\colon \exists k \in K\colon E(x,k) = x'\}| \neq O(p(|x'|))$$

for any polynomial p, and

4) If $E(x,k_0) = x'$ for a particular key k_0, then

$$|\{k\colon E(x,k) = x'\}| = O(q(|x'|))$$

for some polynomial q.

Conditions 1 and 2 ensure that the encryption scheme is feasible; in fact condition 2 eliminates the need to do any decryption $f(x') \longrightarrow f(x)$. The moral of Example 2 is embodied in condition 3, which says that the number of cleartext instances in the preimage of a particular encrypted instances x' is superpolynomial is the size of the instances. Condition 4 is a technical requirement for security: Say $x' \in \mathit{Range}(E)$, $|x'| = n$, and $\{x_0, \ldots x_{2^{n-1}}\}$ is the complete set of cleartext instances in its preimage. If $\{k_1, \ldots, k_{2^n}\}$ is a complete set of the keys that can result in the encrypted instance x', $E(x_0, k_i) = x'$ for $2^{n-1} + 1 \leq i \leq 2^n$, $E(x_i, k_i) = x'$ for $1 \leq i \leq 2^{n-1}$, and $E(x_i, k_j) \neq x'$ for $1 \leq i \leq 2^{n-1}$ and $i \neq j$, then there are a superpolynomial number of preimages of x', but they are extremely unequally probable. If A draws keys uniformly from $\{k_1, \ldots k_{2^n}\}$ and happens to wind up with encrypted instance x', then the probability is at least $\frac{1}{2}$ that she started with cleartext instance x_0. This is not possible if condition 4 is satisfied. Note that there is no requirement that the function E be surjective.

Example 3 is a scheme for encrypting instances of the Comparative Vector Inequalities (CVI) problem; it is clear that the scheme satisfies conditions 1, 3, and 4, so the proof is omitted. Plaisted showed that CVI is NP-complete [P].

Example 3: Each instance of CVI consists of two sets of m-tuples of integers $\{\overline{x_1}, \ldots, \overline{x_k}\}$ and $\{\overline{y_1}, \ldots, \overline{y_l}\}$ about which A asks whether there is an m-tuple $\overline{z}$ such that the number of $\overline{x_i}$ satisfying $\overline{x_i} \geq \overline{z}$ is strictly greater than the number of $\overline{y_j}$ satisfying $\overline{y_j} \geq \overline{z}$, where $\overline{u} \geq \overline{v}$ if and only if no component of $\overline{u}$ is less than the corresponding component of $\overline{v}$. To encrypt an instance, A chooses an element $\overline{w} \in Z^m$ and computes

$$E(\{\overline{x_1}, \ldots, \overline{x_k}\},\ \{\overline{y_1}, \ldots, \overline{y_l}\},\ \overline{w}) = \{\overline{x_1} + \overline{w}, \ldots, \overline{x_k} + \overline{w}\},\ \{\overline{y_1} + \overline{w}, \ldots, \overline{y_l} + \overline{w}\}.$$

Then $\overline{z}$ satisfies $|\{\overline{x_i}\colon \overline{x_i} \geq \overline{z}\}| > |\{\overline{y_j}\colon \overline{y_j} \geq \overline{z}\}|$ if and only if $\overline{z'} = \overline{z} + \overline{w}$ satisfies $\{\overline{x'_i}\colon \overline{x'_i} \geq \overline{z'}\}| > |\{\overline{y'_j}\colon \overline{y'_j} \geq \overline{z'}\}|$, where $\overline{x'_i} = \overline{x_i} + \overline{w}$ and $\overline{y'_j} = \overline{y_j} + \overline{w}$.

Having shown that *one* NP-complete problem admits an encryption scheme satisfying Definition 1, we cannot avoid asking whether they all do. For each NP-complete problem f, there is a polynomial-time reduction r of f to CVI that takes yes-instances to yes-instances and no-instances to no-instances. Can A encrypt an instance x of f by first applying r and then applying the encryption function E from Example 3 to $r(x)$? Not necessarily: The fact that r may not be surjective prevents us from proving that the mapping Eor satisfies conditions 3 and 4 of Definition 1.

If r were truly a *structure preserving*, polynomial-time computable mapping, then E could be composed with it to yield an encryption function for f. Berman and Hartmanis consider such a class of mappings, the p-isomorphisms, in [BH]. We will restate one of their results and then use it to prove something general about the encryptability of NP-complete problems. Let Σ and Γ be two alphabets, C a subset of Σ^*, and D a subset of Γ^*. A p-reduction of C to D is a transducer $T : \Sigma^* \rightarrow \Gamma^*$ that runs in polynomial time such that $T(x) \in D$ if and only if $x \in C$. A p-isomorphism is a *bijection* $f : \Sigma^* \rightarrow \Gamma^*$ such that f is a p-reduction of C to D and f^{-1} is a p-reduction of D to C. Note that the functions f and f^{-1} run in polynomial time on $\Sigma^* \setminus C$ and $\Gamma^* \setminus D$ as well as on C and D.

Theorem (Berman and Hartmanis): An NP-complete set U is p-isomorphic to CNF-SAT if and only if there exist two p-time computable functions S_U and D_U such that

(i) $(\forall x, y)\ [S_U(x, y) \in U \text{ iff } x \in U]$,

(ii) $(\forall x, y)\ [D_U(S_U(x, y)) = y]$.

It is straightforward to find appropriate functions S and D for the CVI problem of Example 3.

Lemma 2: CVI is p-isomorphic to CNF-SAT.

Proof: Suppose $(X = \{\overline{x_1}, \ldots, \overline{x_k}\},\ Y = \{\overline{y_1}, \ldots, \overline{y_l}\})$ is an instance of CVI,

where $\overline{x_i} = (x_{i1}, \ldots, x_{im})$ and $\overline{y_j} = (y_{j1}, \ldots, y_{jm})$. Then put

$$S_{\mathrm{CVI}}((X,Y),v) = (\{x'_1 = (x_{11}, \ldots, x_{1m}, v), \ldots, x'_k = (x_{k1}, \ldots, x_{km}, v)\}, \{y'_1 = (y_{11}, \ldots, y_{1m}, v), \ldots, y'_l = (y_{l1}, \ldots, y_{lm}, v)\}).$$

We have $S_{\mathrm{CVI}}(X,Y,v) \in \mathrm{CVI}$ if and only if $(X,Y) \in \mathrm{CVI}$ because $(z_1, \ldots, z_m)$ is less than or equal to more $\overline{x_i}$'s than $\overline{y_j}$'s if and only if $(z_1, \ldots, z_m, v)$ is less than or equal to more $\overline{x'_i}$'s than $\overline{y'_j}$'s. The obvious algorithm for D_{CVI} (scan $\overline{x'_1}$ and return its rightmost component) gives us $D_{\mathrm{CVI}}(S_{\mathrm{CVI}}((X,Y),w)) = w$. ∎

Berman and Hartmanis state that they know of no NP-complete problems that are not p-isomorphic to CNF-SAT. In particular, they show that CLIQUE is p-isomorphic to CNF-SAT. The question of how many p-isomorphism classes there are among the NP-complete problems remains open, but Mahaney subsequently proved that the number is either one or countably infinite [M]. By Lemma 2, all the NP-complete problems that have been classified are in the same p-isomorphism class as CVI.

Lemma 3: If f is a decision problem that is p-isomorphic to CVI, then f is encryptable under Definition 1.

Proof: Let E_{CVI} be the encryption function for CVI from Example 3, x be an instance of f, and ϕ be a p-isomorphism of f onto CVI. Then the function

$$E_f(x,w) = \phi^{-1}(E_{\mathrm{CVI}}(\phi(x),w))$$

is an encryption function for f.

Because E_{CVI}, ϕ, and ϕ^{-1} run in polynomial time, take yes-instances to yes-instances, and take no-instances to no-instances, E_f does as well; thus E_f satisfies conditions 1 and 2 of Definition 1. If $x' \in \mathit{Range}(E_f)$, then $\phi(x') \in \mathit{Range}(E_{\mathrm{CVI}})$; each instance in the preimage of $\phi(x')$ under E_{CVI} is of the form $\phi(x)$ for a *unique* instance x of f, because ϕ is a bijection. Thus $|\{x\colon \exists w\colon E_f(x,w) = x'\}| = |\{\phi(x)\colon \exists w\colon E_{\mathrm{CVI}}(\phi(x),w) = \phi(x')\}|$ is not $O(p(|x'|))$ for any polynomial p. (Actually, we see immediately that it is not $O(p(|\phi(x')|))$ for any polynomial p, but this is equivalent, because ϕ can cause only polynomial growth or shrinkage in the size of both yes- and no-instances [M].) This means that E_f satisfies condition 3. The proof that it satisfies condition 4 is almost identical. ∎

The following theorem goes as far as we can go by combining Lemmas 2 and 3 with the results of [BH]. No definitive statement can be made about which NP-complete problems are encryptable under Definition 1 without settling the question of whether they are all p-isomorphic.

Theorem 1: All problems that are in the same p-isomorphism class as CNF-SAT are encryptable under Definition 1. No NP-complete problems are known to lie outside of this class.

Finally, we need to exhibit an encryption scheme for the discrete logarithm problem that satisfies Definition 1. First observe that it is possible to pose the problem in yes/no form. For fixed p and g, each instance is a pair $(x,\ [a,\ b])$, where the second argument is a subinterval of $[1,\ p-1]$. All arithmetic is done in Z_p; so if $a > b$ in Z, the elements of the subinterval are $a,\ a+1,\ \ldots,\ p-1,\ 1,\ 2,\ \ldots,\ b$. The answer to the instance $(x,\ [a,\ b])$ is "yes" if and only if there is an $e \in [a,\ b]$ such that $g^e \equiv x \bmod p$. Binary search is used to answer an instance of the standard discrete logarithm problem in $O(\log p)$ iterations of the yes/no version: First choose a random element e of $[1,\ p-1]$, set $[a,\ b]$ to $[e,\ e+\frac{p-1}{2}-1 \bmod p-1]$, and set $[A,\ B]$ to $[1,\ p-1]$. Then repeat these steps until the discrete logarithm of x is in hand: Submit the instance $(x,\ [a,\ b])$ to the yes/no version of the algorithm. If the answer is yes, then set $[A,\ B]$ to $[a,\ b]$ and set $[a,\ b]$ to a random subinterval of this new $[A,\ B]$ of size $\frac{|A-B|+1}{2}$. If the answer is "no", then leave $[A,\ B]$ unchanged and set $[a,\ b]$ to its complement in $[A,\ B]$. The logarithm of x results from an affirmative answer to an instance of the decision problem in which $a = b$. In order to encrypt an instance of the standard version, choose a random c as in Example 1, go through the binary search with $g^c x \bmod p$ as the first argument and both endpoints of every interval translated by $c \bmod p-1$, and subtract c from the final answer.

The results we get with Definition 1 are unsatisfactory. It is possible for an encryption scheme to meet the requirements and still be vulnerable to the criticism that it doesn't really hide anything. This is the case in Example 3, where the possible preimages of an encrypted CVI instance are numerous, and they are *syntactically* different, but they have a lot of structure in common.

The encryption schemes we get for other NP-complete problems by applying Theorem 1 are just as weak in this respect, because they come from Example 3 via isomorphisms. In order to get a meaningful definition of encryptability based on the size of the preimages of encrypted instances, we'd have to make precise when we call two instances different enough to count them separately.

Another important shortcoming of this approach is that it gives no hint of how to prove negative results. Our intuition is that many problems of importance in cryptography, e.g. integer factoring, are *not* encryptable and that the right definition would enable us to prove this.

4. Directions of Current and Future Work

The failure of Definition 1 can be restated as follows: we have not said what the secret is. Exactly *what* about the cleartext instance x cannot be inferred from the encrypted instance x' without knowledge of the key k? In Definition 2, we address this question and ignore completely the size of the preimage of x', which was the focus of Definition 1.

Definition 2: Let f be a decision problem. We say that f is encryptable if there are two functions E_1 and E_2 and a set K of keys such that

1) $E_i : Dom(f) \times K \rightarrow Dom(f), \;\; i = 1, 2,$

2) Both E_1 and E_2 are computable in polynomial time, and

3) $E_1(x, k)$ is a yes-instance of f if and only if x is a yes-instance, and $E_2(x, k)$ is a yes-instance if and only if x is a no-instance.

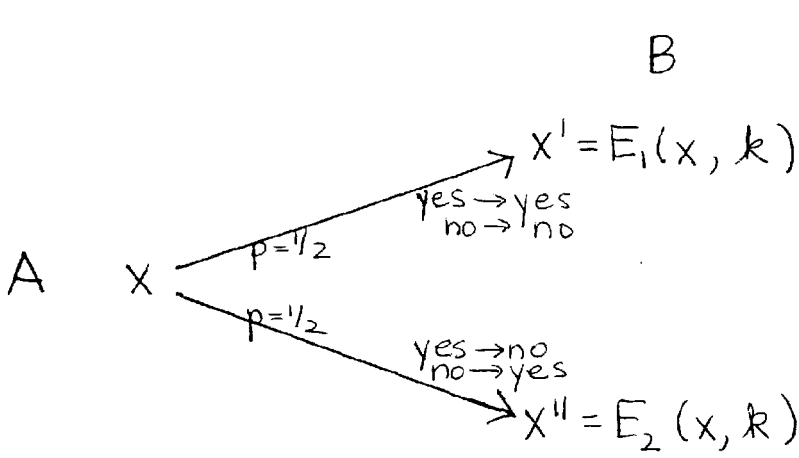

Figure 2: f is encryptable under Definition 2

So A is trying to hide the answer, $f(x)$. She chooses a key and, with probability $\frac{1}{2}$, uses the answer-preserving transformation E_1, with probability $\frac{1}{2}$ the answer-reversing transformation E_2. B tells her "yes" or "no" and has only a 50-50 chance of guessing which is the answer to her original instance.

The first thing to observe about Definition 2 is that it is unlikely to be satisfied by a problem that's NP-complete: the function E_2 would be a polynomial-time reduction from f to its complement and thus could only exist if NP = Co-NP. However, there is a natural example of a decision problem for which such a pair E_1, E_2 can be found:

Example 4: Let $f(n)$, $n \in N^+$, be "yes" if and only if n has an odd number of distinct prime factors. The key-space K consists of small sets of primes. A's algorithm for E_1 is to pick an even number of primes $p_1, \ldots, p_{2t}$ and compute $E_1(x, \{p_1, \ldots, p_{2t}\}) = np_1 \cdots p_{2t}$; to reverse the answer in E_2, she does the same thing using an odd number of primes.

This f belongs to NP $\cap$ Co-NP and leads us to the

Open Question: Is every problem in NP $\cap$ Co-NP encryptable under Definition 2?

Finally, it would be very instructive to find some plausible definition under which we could prove that integer factoring is not encryptable. It is possible that some good would come of an attempt to generalize Definition 2 so that it applied to a broader class of f's than just decision problems.

5. Acknowledgements

I would like to thank my advisor Andy Yao for many helpful discussions of these ideas. I also got a lot of useful feedback from friends at Stanford and at AT&T Bell Labs; special mention is due Eli Upfal for his suggestion that I look at Definition 2.

6. References

[**BH**] Berman, Len and Juris Hartmanis. "On Isomorphisms and Density of NP- and other Complete Sets", *SIAM J. on Comput.*, 6 (2), 1977, 305–322.

[**CF**] Coppersmith, Don and Joan Feigenbaum. "Finite Graphs with Two Inequivalent Factorizations Under the Composition Operator", IBM Research Report RC11149, 1985, submitted to *Journal of Combinatorial Theory, Series B.*

[**FS**] Feigenbaum, Joan and Alejandro A. Schäffer, "Recognizing Composite Graphs is Equivalent to Testing Graph Isomorphism", to appear in *SIAM J. on Comput.*

[**M**] Mahaney, Stephen R. "On the Number of p-isomorphism Classes of NP-complete Sets", Proc. 22nd Annual IEEE Symposium on the Foundations of Computer Science, 22, 1981, 271–278.

[**P**] Plaisted, David. "Some Polynomial and Integer Divisibility Problems are NP-hard", Proc. 17th Annual IEEE Symposium on the Foundations of Computer Science, 17, 1976, 264–267.