This implies that B has a simple eigenvalue 10 and just two further eigenvalues 1 and -5. From trace $(B)=0$ it follows that their multiplicities are respectively 20 and 6. Let $A=I-B$, and $G=G(A)$. Clearly the complement of G is the Schläfli graph, and rank $(A)=7$. Thus $\Theta(G)\leqslant 7$. Applying [1, theorem 9] to G and its complement $\bar{G}$ yields

$$\vartheta(G)\leqslant 9; \quad \vartheta(\bar{G})\leqslant 3.$$

By [1, corollary 2] we have

$$\vartheta(G)\vartheta(\bar{G})\geqslant 27.$$

Hence $\vartheta(G)=9$, and $\vartheta(\bar{G})=3$ (in fact $\Theta(\bar{G})=3$, because G has triangles). So we have $\Theta(G)\neq\vartheta(G)$, $\Theta(G)\cdot\Theta(\bar{G})\leqslant 21<27$, and since $\Theta(G\cdot\bar{G})\geqslant 27$, the questions of Problems 1, 2, and 3 of [1] are answered in the negative.

References

[1] L. Lovász, "On the Shannon capacity of a graph," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 1–7, Jan. 1979.

[2] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities*. Boston: Allyn and Bacon, 1964.

[3] J. J. Seidel, "Strongly regular graphs with $(-1,1,0)$-adjacency matrix having eigenvalue 3," *Linear Algebra Appl.*, vol. 1, pp. 281–298, 1968.

A Note on the Complexity of Cryptography

GILLES BRASSARD

***Abstract*—Evidence is given for the difficulty of an eventual proof of computational security for cryptosystems based on one-way functions, such as the one proposed by Diffie and Hellman. A proof of NP-completeness for the cryptanalytic effort would imply NP = CoNP.**

I. Introduction

Diffie and Hellman [1] have proposed the use of the exponential function in a finite field for cryptographic purposes. This proposal is based on the conjecture that the inverse function, the logarithm, is not feasibly computable. The next best thing to a proof of this conjecture would be to show that the logarithm is **NP**-hard in the sense that there is a Turing machine using it as oracle that can accept some **NP**-complete set in polynomial time. If this were the case, we would know that the logarithm is hard indeed under the generally accepted assumption that $\boldsymbol{P}\neq\mathbf{NP}$ [2, ch. 10].

A word of caution is necessary here. Conventional complexity deals with worst case behavior: a function is said to be not feasibly computable if, for any algorithm A that computes it and any polynomial P, there is at least one input x on which A takes more than $P(n)$ steps, where n is the length of x. That is not quite the notion needed in cryptography: it would be a poor cryptosystem that allows easy enemy decipherment of all but a few cryptograms. Nevertheless, it appears that even a proof that the logarithm is hard in the worst case escapes current techniques, let alone a proof of the computational security of Diffie and Hellman's proposed cryptosystem.

A nondeterministic Turing machine M is said to *compute* the function f if, for any input x, at least one path of the computation halts and if every halting path does so with $f(x)$ on its output tape. Similarly M *accepts* a set S if, for any input x, there exists a computation path which halts (i.e., accepts) if and only if $x\in S$. The key observation is that it is possible to compute finite field logarithms in polynomial time using such a machine. As an application of a theorem by Miller [3], the projection $P_{\log}$ of the logarithm function (defined below) belongs to $\mathbf{NP}\cap\mathbf{CoNP}$. Moreover, $P_{\log}\in\boldsymbol{P}$ if and only if the logarithm can be computed in deterministic polynomial time. A proof of Diffie and Hellman's conjecture would therefore imply that $\boldsymbol{P}\subsetneq\mathbf{NP}\cap\mathbf{CoNP}$ with $P_{\log}$ as witness. Furthermore, should the logarithm be **NP**-hard, we would conclude that **NP** = **CoNP** because $P_{\log}$ would be both **NP**-complete and a member of **CoNP**. More details follow. Since both these results ($\boldsymbol{P}\subsetneq\mathbf{NP}\cap\mathbf{CoNP}$ and **NP** = **CoNP**) are expected to be either false or very difficult to establish, we conclude this is also the case for the security of the cryptosystem.

Manuscript received April 28, 1978; revised September 11, 1978. This work was supported by the IBM Thomas J. Watson Research Center.

The author is with the Department of Computer Science, Cornell University, Ithaca NY 14850.

II. Construction

In an attempt to make this section self-contained, we shall not use Miller's result [3] explicitly. The techniques used will however be similar to his. We shall also short-circuit the proof that the logarithm can be computed in polynomial time by a nondeterministic Turing machine.

Define the logarithm function $\log(p,r,n)$ as follows: if p is a prime, r is a primitive root of p [4, sec. 4.5.4], and $0<n<p$, then $\log(p,r,n)$ is the unique m such that $0<m<p$ and $r^m=n \pmod p$. Otherwise $\log(p,r,n)$ is zero. Define the *projection* $P_{\log}$ as

$$\{\langle p,r,n,t\rangle \mid \log(p,r,n)>t\}$$

where $\langle p,r,n,t\rangle$ is an abbreviation for $\langle p,\langle r,\langle n,t\rangle\rangle\rangle$ and $\langle\cdot,\cdot\rangle$ is a suitable pairing function.

$P_{\log}\in\mathbf{NP}$: p is a prime and r is a primitive root of p if and only if $r^{p-1}=1 \pmod p$ and for each q a prime factor of $p-1$, $r^{(p-1)/q}\neq 1 \pmod p$ [4, sec. 4.5.4]. These conditions can be checked in nondeterministic polynomial time by guessing the prime decomposition of $p-1$ together with certificates of primality [5] for each of the factors and by using a divide-and-conquer strategy to compute exponentials modulo p [4, sec. 4.6.3]. Once it is known that p is prime and r is a primitive root, assuming $0<n<p$, the unique m, $0<m<p$, such that $r^m=n \pmod p$ can be guessed. If $m>t$, $\langle p,r,n,t\rangle$ is in $P_{\log}$.

$P_{\log}\in\mathbf{CoNP}$: $\langle p,r,n,t\rangle$ is not in $P_{\log}$ if and only if $\log(p,r,n)\leqslant t$. This is true if and only if either there are i and j $(0<i<j<p)$ such that $r^i=r^j \pmod p$, there is an $m\leqslant t$ such that $r^m=n \pmod p$, or it is not the case that $0<n<p$. These conditions can be checked nondeterministically in polynomial time.

If $P_{\log}\in\boldsymbol{P}$, then the logarithm can be computed deterministically in polynomial time by the following divide-and-conquer algorithm:

```
function log (p,r,n)
  i := 0
  j := p-1
  while i<j do
    /* i ≤ log ≤ j */
    k := ⌊(i+j)/2⌋
    if ⟨p,r,n,k⟩ ∈ P_log then i := k+1
                         else j := k fi
  od
return i
end log.
```

Finally, should the logarithm be **NP**-hard, then so would $P_{\log}$, because the above algorithm shows that one can compute the logarithm efficiently given $P_{\log}$. An oracle for $P_{\log}$ can therefore be used to simulate an oracle for the logarithm. $P_{\log}$, being in **NP**, would be **NP**-complete. However, $P_{\log}\in\mathbf{CoNP}$. We conclude that **NP** = **CoNP**, for otherwise there could be no **NP**-complete sets the complement of which are in **NP**. We give a proof of this claim in the Appendix.

0018-9448/79/0300-0232$00.75

III. Generalization

The reasoning above can be generalized to other cryptosystems based on one-way functions. A one-way function, as defined in [1], is a function f such that, for any argument x in the domain of f, it is easy to compute the corresponding value $f(x)$, yet, for almost all y in the range of f, it is computationally infeasible to solve the equation $y=f(x)$ for any suitable argument x. So far we have used a candidate for one-way functions set forth by Diffie and Hellman which is the exponentiation in a finite field.

It turns out that if any function f, satisfying a few restrictions, can be proved to be one-way, then $P \not\subseteq \mathbf{NP} \cap \mathbf{CoNP}$. Similarly, if evidence is given that f is one-way by showing its inverse to be **NP**-hard, then $\mathbf{NP}=\mathbf{CoNP}$. The restrictions are that f be one-one from an **NP** domain onto a **CoNP** range and that there be a polynomial P such that, for any x in the range of f, $|x| \leq P(|f(x)|)$, where $|x|$ denotes the length of the binary representation of x.

This leads us to a final example. Rivest, Shamir, and Adleman [6] have proposed a public-key cryptosystem based on the belief that the function *multiplication* is one-way when restricted to prime numbers. In more conventional terms, it is easy to multiply two prime numbers, but we as yet do not know how to efficiently find the factors of a large number even with the foreknowledge that it is the product of two primes. Here again it turns out that a proof of computational security for their cryptosystem would imply that $P \not\subseteq \mathbf{NP} \cap \mathbf{CoNP}$ while a proof of **NP**-hardness for the cryptanalytic effort would imply that $\mathbf{NP}=\mathbf{CoNP}$. For a direct proof of these results, mimic the proof of Section II replacing $P_{\log}$ by S, which is defined as

$$\{\langle x,y\rangle | x \text{ has a prime factor smaller than } y\}.$$

As in Section II, one proves that $S \in \mathbf{NP} \cap \mathbf{CoNP}$ and that $S \in P$ if and only if prime decomposition is feasibly computable.

Acknowledgment

Similar results have been found independently by Leonard Adleman, Ronald Rivest, and Gary Miller [7].

Appendix

We shall prove that if a set A is **NP**-complete and if $A \in \mathbf{CoNP}$, then $\mathbf{NP}=\mathbf{CoNP}$. This is obvious for Karp's notion of **NP**-completeness [8] but requires proof for Cook's notion [9], which we have used throughout this correspondence. Let $M+$ and $M-$ be nondeterministic polynomial-time-bounded Turing machines that accept A and its complement, respectively.

For any $L \in \mathbf{NP}$, there is a deterministic polynomial-time-bounded Turing machine M which accepts L given an oracle for A. Consider the nondeterministic Turing machine M', which, on any input, deterministically simulates M step by step except when M asks a question of the oracle. In such a case, M' guesses whether the answer ought to be "yes" or "no" and nondeterministically simulates either $M+$ or $M-$ depending on this choice. Should M' reach an accepting state of $M+$ or $M-$, it would know it is on the right track and would resume the deterministic simulation of M in the appropriate oracle answer state. Should M' reach a rejecting state of $M+$ or $M-$ instead, it would know nothing at all and would immediately reject the original input. Finally, when M reaches a final state, M' accepts if and only if M rejects.

It is obvious that M' runs in polynomial time and that it accepts the complement of L, so that $L \in \mathbf{CoNP}$. This proves that $\mathbf{NP} \subseteq \mathbf{CoNP}$. Now, given any $L \in \mathbf{CoNP}$, its complement is in **NP** by definition of **CoNP** and hence in **CoNP**, since $\mathbf{NP} \subseteq \mathbf{CoNP}$. $L \in \mathbf{NP}$ again by definition of **CoNP**. This shows that $\mathbf{CoNP} \subseteq \mathbf{NP}$ and completes the proof that $\mathbf{NP}=\mathbf{CoNP}$.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.
[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[3] G. L. Miller, "Riemann's hypothesis and tests for primality," *J. Comput. Syst. Sci.*, vol. 13, pp. 300–317, 1976.
[4] D. E. Knuth, *The Art of Computer Programming*, vol. 2, *Semi-Numerical Algorithms*. Reading, MA: Addison-Wesley, 1969.
[5] V. Pratt, "Every prime has a succinct certificate," *SIAM J. Comput.*, vol. 4, pp. 214–220, 1975.
[6] R. Rivest, A. Shamir, and L. Adleman, "A method of obtaining digital signatures and public-key cryptosystem," *Commun. Ass. Comput. Mach.*, vol. 21, pp. 120–126, Feb. 1978.
[7] L. Adleman, private communication, 1978.
[8] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. NY: Plenum, 1972.
[9] S. A. Cook, "The complexity of theorem proving procedures," in *Proc. 3rd Ann. ACM Symp. Theory of Computing*, 1971, pp. 151–158.

Repeated Use of Codes which Detect Deception

VIIVEKE FÅK

***Abstract*—A sender A wants to send N messages x_i, $i=1,\cdots,N$, chosen from a set containing M different possible messages, $M>N$, to a receiver B. Every x_i has to pass through the hands of a dishonest messenger C. Therefore A and B agree on a mathematical transformation f and a secret parameter, or key k, that will be used to produce the authenticator $y_i=f(x_i,k)$, which is sent together with x_i. The key is chosen at random from a set of L elements. C knows f and can find all elements in the set $G(x_i,y_i)=\{k|f(x_i,k)=y_i\}$ given enough time and computer resources. C wants to change x_i into x' without B suspecting. This means that C must find the new authenticator $y'=f(x',k)$. Since $G(x_i,y_i)$ can be found for any (x_i,y_i), it is obvious that C will always succeed unless $G(x_i,y_i)$ contains more than one element. Here it is proved that the average probability of success for C is minimized if (a) $G(x_i,y_i)$ contains $L^{(N-1)/N}$ elements and (b) each new known pair (x_j,y_j) will diminish this set of solutions by a factor of $L^{-1/N}$. The minimum average probability will then be $L^{-1/N}$.**

In [1], Gilbert, MacWilliams, and Sloane analyzed the situation in which sender A wants to pass one message x_i to receiver B via the dishonest messenger C. The message x_i is accompanied by an authenticator y_i, which is a function f of x_i and a secret key k, which is chosen at random from a set of L elements. In this case C can find all values $k \in G(x_i,y_i)=\{k|f(x_i,k)=y_i\}$. Since C also can compute $y'=f(x',k)$ for any x' and any k, he can substitute x' for x_i without detection if he uses a key that yields the same y' as the correct key. If all keys in the set $G(x_i,y_i)$ yield the same $y'=f(x',k)$, C would always succeed. In [1], it is shown that C will have a minimum probability of success if all $k \in G(x_i,y_i)$ give different y_j for every x_j, $j \neq i$. Furthermore, it is shown that this minimum probability is $L^{-1/2}$. To obtain that limit, y, k, and x should be samples of stochastic variables with rectangular probability distributions, and the set $G(x_i,y_i)$ should contain $L^{1/2}$ elements for every obtainable pair (x_i,y_i). But with this strategy, a second message with the same key would disclose the key, since only one key fits both $k \in G(x_i,y_i)$ and $k \in G(x_j,y_j)$, $i \neq j$. The strategy and its limits are used in [1] to construct ideal codes for sending one authenticated message. In the present correspondence, we study how codes, i.e., authenticating functions f, should be constructed in order to minimize the risk of deception when k is used several times.

Manuscript received January 24, 1978; revised May 24, 1978.
The author is with the Department of Electrical Engineering, Linköping University, Linköping, S-581 83, Sweden.

0018-9448/79/0300-0233$00.75