A PUBLIC-KEY CRYPTOSYSTEM BASED ON THE WORD PROBLEM

Neal R. Wagner[1]
Marianne R. Magyarik

Drexel University
Mathematics and Computer Science
Philadelphia, Pennsylvania 19104

ABSTRACT.

The undecidable word problem for groups and semigroups is investigated as a basis for a public-key cryptosystem. A specific approach is discussed along with the results of an experimental implementation. This approach does not give a provably secure or practical system, but shows the type of cryptosystem that could be constructed around the word problem. This cryptosystem is randomized, with infinitely many ciphertexts corresponding to each plaintext.

1. NP-COMPLETE PROBLEMS.

The idea of using an NP-complete problem to construct a public-key cryptosystem (PKC) seemed promising [Diff76], but has not been successful historically. The earliest such PKC was based on the integer knapsack problem, and recently various versions of this PKC have been broken by general, powerful attacks [Sha83a], [Adle83]. (In this case, the attacks have been carried out on the type of trapdoor inserted, and not directly on the

[1] This work was supported in part by NSF grant DCR-8403350, and by Drexel University's Faculty Development Mini-Grant program

knapsack problem itself.) Other PKC's based on NP-complete problems have been proposed, but none seems successful so far.

There is a hierarchy of decision problems, from simplest to hardest, extending from polynomial-time problems, through NP-complete problems, to the undecidable problems (the hardest of all) [Tarj83], [Aho74].

NP-complete problems are often regarded as lying on the "boundary" of intractability, i.e., they are the simplest known "natural" problems which are intractable. Stated differently, if an NP-complete problem is even slightly weakened, it may no longer be intractable. In constructing a PKC, one must insert a trapdoor, and after allowing for all the various kinds of attacks, we would not usually expect to have a "pure" NP-complete problem remaining.

Along similar lines Brassard [Bras79] has shown that, with a few restrictions, if a cryptosystem were provably NP-complete to break, then the theoretical result NP= co-NP would follow. This latter result is widely conjectured to be untrue, though no proof is available [Gary79]. Thus the cryptanalysis of a PKC based on an NP-complete problem would be easier than NP-complete, hence likely a tractable computation.

There is a large body of theory about NP-completeness, but the theory only applies to worst-case analyses and to arbitrarily large problem instances. For example, the integer knapsack problem is not *strong* NP-complete, meaning that polynomial-time algorithms are available unless "exponentially large" integers are used in the problem instance [Gary79]. (Problems that are not strong in this sense are said to be solvable in *pseudo-polynomial time*.)

Two other classes of polynomial-time algorithms can be used to try to solve NP-complete problems. There are clever *approximation* algorithms which always get a approximate answer, though not necessarily the exact answer. (See [Gary79] and [Horo78] for examples.) There are also *non-deterministic* algorithms, which give an exact answer but may not give any answer at all [Horo78]. Of course there is no known polynomial-time algorithm that will solve worst-case, arbitrarily large problem instances, but algorithms like those above might force unacceptably large instances of an NP-complete problem if a PKC using it is to be secure.

There is specialized problem, *factoring,* that has become the basis for several cryptographic applications, including

- the RSA-like cryptosystem with exponent 2 [Rabi79], [Will80],
- the generation of a cryptographically secure random number generator [Sha83c], and
- the exchange of secret keys without an arbiter [Blum83].

Each has a protocol with a complete security proof, assuming that factoring is intractable. And of course the RSA cryptosystem itself, while not equivalent to factoring, depends on the difficulty of factoring for its security [Rive79]. New applications that depend on factoring appear regularly [Ong84]. As long as factoring remains intractable, we are in a good position, but we are overdependent on the computational complexity of one particular problem.

The exact complexity status of the factoring problem is not known, though as with knapsack, factoring is easily solved in pseudo-polynomial time. As before, even if no polynomial-time algorithm is found, unacceptably large integers might eventually be required to keep the problem intractable.

* * * *

Thus it seems natural and desirable to look toward harder problems as the basis for a PKC. There are various *provably intractable* problems [Aho74], and of course the undecidable problems for which no general algorithmic solution can exist. It is important to note that for a PKC one could use only a *special instance* of one of these harder problems. The difficulty of cryptanalysis would still be in the class NP.

This paper is the result of an initial look at various undecidable problems, trying to construct PKC's. We are concentrating on a particular problem along the lines of an earlier paper [Wagn84], with more specific details included here.

2. THE WORD PROBLEM.

There are undecidable problems for finitely presented groups and for semigroups. First we need a number of definitions. (See [Magn66], [Rotm73], [Crow63], [Lynd77].) A

finitely presented group G consists of *generators* $x_1, x_2, \ldots, x_n$, which are just abstract symbols, and *relators* $r_1 = e, r_2 = e, \ldots, r_m = e$, to be defined below. Corresponding to each generator x_i there is an *inverse* x_i^{-1}. A *word* in G is a finite string made up of symbols x_i and x_i^{-1}. The *empty string* e is also a word, the *identity* of the group. Each of the r_i above is a word. The group operation for combining words is *concatenation*. For each word w, the *inverse word* w^{-1} consists of all the symbols of w written in reverse order, where each x_i is replaced by x_i^{-1} and each x_i^{-1} is replaced by x_i.

The group G consists of equivalence classes of all possible words. Two words w and v are *equivalent* in G if we can transform w to v by a finite sequence of *replacement rules* of the form

Rule (i): changing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ to e, that is eliminating $x_i x_i^{-1}$ or $x_i^{-1} x_i$,
Rule (ii): introducing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ at any point,
Rule (iii): changing r_j or r_j^{-1} to e, that is eliminating r_j or r_j^{-1},
Rule (iv): introducing r_j or r_j^{-1} at any point.

There is a more formal way to define these concepts. First the *free group* F on generators $x_1, x_2, \ldots, x_n$ is defined as the set of all words in the x_i and x_i^{-1} that are *reduced* by repeatedly cancelling out $x_i x_i^{-1}$ and $x_i^{-1} x_i$ until no further cancellations are possible. Let R be the *normal* subgroup generated by the words $r_1, r_2, \ldots, r_m$. (R is the intersection of all normal subgroups containing the r_i.) Finally G is the quotient group F/R.

The *word problem* for a group G is the decision problem that asks for each word w, whether w is equivalent to the identity of G. (Equivalently one can ask whether two given words are equivalent.) It turns out that there exist *specific* groups for which the word problem is undecidable [Novi55], [Boon59], [Rabi58]. Like any

undecidable problem, the word problem can only be undecidable as a question asked about infinitely many words -- any finite collection of words must have a decidable word problem.

Finitely presented groups are extremely complex objects. For example, the free group on two generators with no relators contains within it as a subgroup the free group on a countably infinite number of generators. There is a great deal of structure and theory associated with such groups and with the word problem. A number of researchers devote their entire energies to this subject [Lynd77].

There is a similar and simpler *word problem* for *semigroups*. We start with generators and words in the generators as before but without the inverses. Instead of relators we have a list of *equations* of the form $a_1 = b_1$, $a_2 = b_2$, ..., $a_m = b_m$. In defining equivalent words we can only replace any occurrence of a_i by b_i and vice versa. The word problem for semigroups again asks if we can decide whether two given words are equivalent. Using the halting problem, for example, it is easy to see that there is a specific semigroup for which the word problem is undecidable.

All of our discussion of groups in this paper can be regarded as just a special case of semigroups since one could regard the group as a semigroup with extra symbols x_i^{-1} and extra equations $x_i x_i^{-1} = e$, etc. Thus a semigroup would just be more general and flexible for our applications. We have chosen to emphasize groups because they seem to fit in naturally with our main ideas for cryptosystems and because of the enormous amount of research on groups and their word problems. One hopes that such a thoroughly studied problem might someday yield a good theoretical foundation for a cryptosystem.

3. PUBLIC-KEY CRYPTOSYSTEMS.

The word problem is similar to the knapsack problem in that both are "natural" problems for public-key cryptosystems, i.e., both immediately and directly allow public encryption. The difficulty is to insert a trapdoor that will allow decryption. (See [Diff76].)

The trapdoor then becomes a point of weakness for cryptanalytic attacks. We feel that a harder problem may make direct attacks more difficult and allow more leeway for such trapdoor insertions.

To use the word problem to encrypt a *single bit*, start with a finitely presented group G and with two "special" words w_1 and w_2 known to be inequivalent in G. Choose one of w_1 and w_2 and randomly apply Rule (i) through Rule (iv) to the word, resulting in a word v equivalent to either w_1 or w_2 (but not both). Thus the public key consists of the group G and the special words w_1 and w_2.

This is a randomized encryption procedure in the sense of [Rive83]. There are infinitely many possible ciphertexts corresponding to each plaintext *bit*, and the system has an arbitrarily large expansion factor.

This property of a large expansion factor is not new. In fact the homophonic ciphers introduced centuries ago [Denn82] associate one of a (finite) set of ciphertext elements with each plaintext element. In this case security is increased with greater expansion factors to perfect security "when each letter of plaintext enciphers into a unique ciphertext symbol" [Denn82]. As another example Brassard [Bras81] mentions a message length n to ciphertext length n^2 expansion. More generally various randomization techniques [Rive82] can be used to trade a larger expansion factor for the likelihood of increased security.

One can improve the large expansion factor in our system as follows: to encrypt n bits, select $q = 2^n$ mutually inequivalent special words $w_1, w_2, \ldots, w_q$. Choose one of these and encrypt as above. This gives an n-fold improvement in expansion factor, but makes the "special word" part of the public key 2^n times as long.

With good choices for the group and special words in the encryption method described above, it appears that decryption can be made very difficult. Decryption difficulty also depends on the number and type of replacements made during encryption.

4. TRAPDOOR INSERTION.

There are surely many ways to insert a trapdoor. We present one general approach in section 4.1. Section 4.2 discusses cryptanalysis, and section 4.3 gives a more specific approach whose implementation is discussed in section 5.

4.1. GENERALITIES.

Start with a finitely presented group

$$G = (x_1, x_2, ..., x_n | r_1 = e, r_2 = e, ..., r_m = e),$$

and add more relators

$$s_1 = e, s_2 = e, ..., s_p = e,$$

to get another finitely presented group G'. There is a special relationship between G and G'. In formal group theory terms if N is the normal subgroup of G generated by the words $s_1, s_2, ..., s_p$, then G' is the *quotient group* $G' = G/N$. There is a natural function (the *quotient mapping*) $\Omega: G \dashrightarrow G'$ defined as follows. If x is a group element of G (= equivalence class of words) let w be any word representing x. Then $\Omega(x)$ is just the equivalence class of w within G'. For us the most important property of Ω is the following: if x and y are equivalent in G, then $\Omega(x)$ and $\Omega(y)$ are equivalent in G'. Stated another way, if $\Omega(x)$ and $\Omega(y)$ are *not* equivalent in G', then x and y must be not equivalent in G. (The converse does not hold: in fact it is easy for elements *not* equivalent in G to "collapse" to equivalent elements in G'.)

For this trapdoor to work, the w_1 and w_2 from G that are part of the public key must have the property that $\Omega(w_1)$ and $\Omega(w_2)$ are not equivalent in G'. To decrypt one needs to decide in G' which of $\Omega(w_1)$ and $\Omega(w_2)$ the word $\Omega(y)$ is equivalent to, i.e., one needs to solve the word problem in G', at least for some words. ($\Omega(y)$ must be equivalent in G' to one or the other, but not both.)

The idea behind this method is that the word problem in G might be intractable, while the extra relators $\{s_i = e\}$ might simplify things so that there is an efficiently solvable word problem for G'. This general decryption method would work both for the owner of the PKC and for an opponent attempting cryptanalysis. This method is a standard way in group theory to show that two elements are not equivalent.

4.2. CRYPTANALYSIS.

We now list possible cryptanalytic attacks on this type of cryptosystem. Assume that a finitely presented group G is given along with two special inequivalent words w_1 and w_2 for the public key. We regard the quotient group G' or the extra $s_i = e$ relators as the secret key. One of w_1 or w_2 is chosen and encrypted to form a word y.

Attack (a): *Find a tractable algorithm which decides the word problem in G.* With this algorithm, one directly decides which of w_1 or w_2 is equivalent to y. Gilles Brassard has pointed out that there is always a simple but impractical constructive algorithm that works for two words (or finitely many words): just try all possible sequences of replacements in parallel on w_1 and w_2, producing the word y in a finite amount of time. Thus as we have mentioned before, in no sense is cryptanalysis undecidable. However, one hopes that with a good choice of G, there will be no tractable algorithms for direct attacks of this sort.

Attack (b): *Find extra relators* $\{s_i = e\}$ *with quotient group* G', *so that* $\Omega(w_1)$ *is not equivalent to* $\Omega(w_2)$ *and so that there is a tractable algorithm in* G' *to decide the word problem.* This is just the general method mentioned in section 4.1. These particular extra relators do not need to be the same as in the secret key -- just so the other conditions are satisfied. Guarding against this bothersome attack is the main reason for

the complexity of section 5.

Attack (c): *Use a brute-force attack on a PKC, in which one decrypts the ciphertext under each possible secret key.* This attack would succeed and shows that cryptanalysis is in the class NP. However, unlike the situation with a traditional PKC, here there is no fixed bound on the size of the smallest secret key that would work for decryption. There might be infintely many possible candidate secret keys to try, and success might take arbitrarily long. (The specific system described in section 5 has only finitely many possible secret keys for each public key.)

Attack (d): *Regard the system as a conventional cryptosystem, and use a brute-force known plaintext attack in which the plaintext is encrypted under each possible key.* This attack is almost hopeless, since even with the correct key the ciphertext is not determined, and since there is no bound on the number of keys to try.

4.3. DETAILS.

In order to construct a specific trapdoor, we propose choosing the additional relators $\{s_i = e\}$ so that each of the $r_i = e$ becomes trivial. The words r_i are also chosen to facilitate this. Consider extra relators of one of three forms:

Type (S1): (*Elimination* of a generator)
$x_i = e$, for some specific i. (Thus any occurrence of x_i just drops out.)

Type (S2): (*Collapse* of two generators to one)
$x_i x_j^{-1} = e$, or $x_i x_j = e$, for specific i and j. (Any reference to x_j can be replaced by x_i or by x_i^{-1}.)

Type (S3): (*Commutator* of two generators)
$x_i x_j x_i^{-1} x_j^{-1} = e$, or $x_i x_j = x_j x_i$, for specific i and j. (x_i and x_j commute.)

Such extra relators might greatly simplify the r_i words. In fact we will choose the r_i and the s_j in such a way that each r_i, in conjuction with all the s_j, will reduce

to the empty word e. After eliminating and collapsing generators, the group G' will have *only* relators of type (S3) (the commutators). Thus G' will be a free group in which certain pairs of generators commute. There is a simple (polynomial-time) algorithm which decides the word problem for such a group. (See section 5.) Notice that in G' the special words w_1 and w_2 must still not be equivalent.

This method for inserting a trapdoor could also be as an attack by an opponent, a special case of Attack (b) of the previous section.

Attack (b'): *Find extra relators* $\{s_j = e\}$ *of types* (S1), (S2) *and* (S3) *such that*

(i) *each* r_i *word becomes trivial, and*
(ii) *even with the extra* $\{s_j = e\}$ *relators, the words* w_1 *and* w_2 *are still not equivalent.*

We propose to choose the special words w_1 and w_2 so that for "most" choices of $s_j = e$ relators, condition (ii) will not be true. Here is our approach in outline form. Given a large collection of $r_j = e$ relators the opponent or PKC originator must introduce many commuting pairs of generators in order to make all the r_j trivial. So in the simplified group G', most pairs of generators will commute. The PKC originator will have a small (secret) subset of non-commuting pairs.

It is fairly easy to construct arbitrarily many inequivalent words that reduce to e if any one of a set of pairs commutes. For a single pair (x_1, x_2), just use the word $x_1 x_2 x_1^{-1} x_2^{-1}$ or $x_1 (x_2)^2 x_1^{-1} (x_2^{-1})^2$, etc. For a recursive general definition, assume u is a word that reduces to e in case any one of a set U of pairs commutes. Assume v and V have the same property. Then the word $u v u^{-1} v^{-1}$ will reduce to e in case any one pair commutes from the union of the sets U and V.

In this form, w_1 and w_2 would still not work in a public key. Before publishing them, they must be encrypted as we have described, so that the set of commuting pairs will no longer be recognizable.

5. AN EXPERIMENTAL IMPLEMENTATION.

We have chosen specific parameters, relators, and special words and have written a computer program to implement an example cryptosystem. We should emphasize that we are only attempting a rough implementation to demonstrate the feasibility of this system, and to stimulate further research. Much more work will be required before anyone could rely on the security and practicality of any cryptosystem similar to this one.

The relators $r_j = e$ are chosen so that for each generator x_j appearing in r_j there is a corresponding x_j^{-1}. Then it is easy to make each r_j trivial by allowing certain pairs to commute, i.e., adding relators of type (S3) (section 4.3). Adding relators of types (S1) and (S2) will give alternative ways to make the r_j trivial. The basic idea is to present an opponent with a very large number of ways to get rid of the r_j relators. We might hope that the opponent would have to search for the secret subset of non-commuting pairs in order to break this system.

After some searching around, we have settled on relators of three types for the original $\{r_j = e\}$, where below x_i, x_j, x_k, and x_l stand for arbitrary generators or inverses of generators.

Type (R1): $x_i x_j x_k x_l x_i^{-1} x_k^{-1} x_j^{-1} x_l^{-1} = e$,
Type (R2): $x_i x_j x_k x_i^{-1} x_j^{-1} x_k^{-1} = e$, and
Type (R3): $x_i x_j x_k x_i^{-1} x_k^{-1} x_j^{-1} = e$.

We mostly use relators of type (R1) with some of types (R2) and (R3). We do not know the complexity of the word problem for a group made up of relators of these forms, so that Attack (a) of section 4.2 might succeed. (It would be better to start with a group G with an undecidable word problem.)

Type (R1) has the advantage that there are seven distinct ways to make such a relator vanish using a minimal number of extra relators of types (S2) (*collapsing*) and (S3) (*commutators*). (Type (S1) (*elimination*) relators are rather too drastic to use much, if at all.) For example, suppose we use an extra collapsing relator

$x_i\, x_l = e$, or $x_i = x_l^{-1}$,

and three extra commutators

$x_i\, x_k\, x_i^{-1}\, x_k^{-1} = e$, or $x_i\, x_k = x_k\, x_i$,

$x_j\, x_k\, x_j^{-1}\, x_k^{-1} = e$, or $x_j\, x_k = x_k\, x_j$, and

$x_k\, x_l\, x_k^{-1}\, x_l^{-1} = e$, or $x_k\, x_l = x_l\, x_k$.

The original relator simplifies as follows

$$x_i\, x_j\, x_k\, x_l\, x_i^{-1}\, x_k^{-1}\, x_j^{-1}\, x_l^{-1} =$$

$$x_i\, x_j\, x_k\, x_j^{-1}\, x_i^{-1}\, x_k^{-1}\, x_j^{-1}\, x_j =$$

$$x_i\, x_j\, x_k\, x_j^{-1}\, x_i^{-1}\, x_k^{-1} =$$

$$x_i\, x_j\, x_j^{-1}\, x_k\, x_i^{-1}\, x_k^{-1} = x_i\, x_k\, x_i^{-1}\, x_k^{-1} = e.$$

There are four other very similar distinct methods to make this relator vanish. In additon, just setting $x_i = x_l$ makes everything drop out and making five of the six possible pairs commute also makes the relator vanish.

Along similar lines there are three ways to make a relator of type (R2) vanish and two ways for a relator of type (R3). Of course any of these relators will vanish if one just allows all relevant pairs to commute, with no need to include the pair (x_j, x_k) in types (R1) or (R3).

In making up a specific PKC we have chosen four non-commuting pairs and made up special words (at least 64 symbols long) that would vanish if any one of the four pairs commuted. These pairs were chosen so that for each pair there is a specific $r_j = e$ relator so that all but one way of making the word r_j trivial will also make the given pair commute. Thus if an opponent uses Attack (b') of section 4.3, then in making each r_j vanish, he will very likely make one or more of the crucial pairs commute, and so the special words will also vanish. (In order to keep the special word from degenerating, it was necessary to add extra non-commuting pairs.)

Applications of the replacement rules (i) through (iv) are more complicated than one might expect. For example, suppose we have a relator

$$r_1 = x_1 x_2 x_3 x_4 x_5 x_6 = e$$

and a word

$$w = x_3 x_6 x_1 x_2.$$

Then in the equation $r_1 = e$, multiply on the left by $x_2^{-1} x_1^{-1}$ and on the right by x_6^{-1} to get

$$x_3 x_4 x_5 = x_2^{-1} x_1^{-1} x_6^{-1}.$$

Taking the inverse of both sides gives

$$x_5^{-1} x_4^{-1} x_3^{-1} = x_6 x_1 x_2.$$

Thus in a group with the relator $r_1 = e$, the word $w = x_3 x_6 x_1 x_2$ is equivalent to the word

$$x_3 x_5^{-1} x_4^{-1} x_3^{-1}.$$

(The process given above can be redone using just Rules (i) through (iv) in a formal fashion.)

In general we can write the generators of a relator clockwise in a circle, and any clockwise connected string can be replaced by the inverse of the complementary string, plus inverses of these replacements. The relator of length 6 above allows 72 different possible replacements, and a relator of length n allows $2n^2$. Our computer program attempts to look for replacements where the string being replaced is as long as possible. This kind of string matching can be done fairly efficiently using a variation of the Knuth-Morris-Pratt algorithm [Aho74].

In actual runs of our experimental implementation, we tried $n = 25$ and $n = 50$ generators. (We think the latter size might provide moderate security against attacks we can visualize.) The special words w_1 and w_2 are first made up as described in section 4.3, and then must be "pre-encrypted" before public release to hide the non-commuting pairs used in making them up. Actual public encryption just consists of more of the same kinds of replacements. Table 1 shows sets of parameters for two cryptosystems.

Table 1. Parameters for two experimental cryptosystems.

Number of Generators		25	50
Number of relators in G	Type R1	34	153
	Type R2	6	21
	Type R3	6	20
	Total	46	194
Pairs of generators in G	Total	300	1225
	Non-commut.	19	29
Number of additional relators in G	Type S1	0	0
	Type S2	3	3
	Type S3	220	1067
	Total	223	1070
Length of special word	Original	64	64
	Encrypted	~1/4 symbol per replacem.	~1/2 symbol per replacem.
Public key size (bits)		~1000-2000	~10000
Expansion factor (minimum)		~50-500	~100-1000

The replacements used for public encryption pose interesting problems. There need to be many random choices in the invocation of these replacements, but we do not want things completely random because we want the lengths to stay within reasonable bounds. It is also necessary that all parts of the original word get acted upon. Finally, we do not want a replacement to just undo the action of a previous replacement. To help with these goals, we maintained a "ghost" string in parallel with the real string being encrypted. The ghost keeps track of which string symbols have been replaced, using which relator. The replacement strategy was to choose a string location and relator at random, and to definitely use that relator for a replacement, trying first near the chosen location. But the algorithm was given some leeway to try to achieve the above goals. We performed thousands of replacements on the special words to get an idea of the

asymptotic behavior. With 50 generators, the last of the original symbols of the special word was replaced after about 1000 replacements. The encrypted special words were growing at the rough rate of half a symbol per replacement.

(After 2000 replacements, the special words were about 1100-1200 symbols long, though the rate of increase seemed to be slowing down.) With a better choice of the set of relators $\{r_j = e\}$ or with more of them, encryption might not have a lengthening effect at all. We hope there is a clever way to do the design so that cryptanalysis is provably equivalent to some standard problem from combinatorial group theory, just as other systems have been proven equivalent to factoring (see section 1).

For decryption in G', we need to solve the word problem for a free group with some commutators. The algorithm converts any word to a standard form in two phases. First consider any substring of the form $...xyx^{-1}...$, where x is a generator, y is a string, and x commutes with every generator in y. In such a case we cancel x and x^{-1}. Such cancellations are repeated until no more are possible. (One needs to argue that any choices along the way do not affect the final outcome.) The second part of the algorithm uses a bubblesort-type method to make sure any adjacent commuting pairs are in a standard order.

In constructing these experimental cryptosystems, most relators were just chosen at random *after* deciding on the trapdoor. With more care, one could create the trapdoor after the relators. In this way the public relators could be represented pseudo-randomly, greatly reducing the public key size.

As part of the experiment, we simulated one simple attack by the opponent. For each type R1 relator we made five of the six pairs commute so that the relator would vanish, and similarly for types R2 and R3. Then of course several of the crucial pairs used in the special words commuted, so that these special words just reduced to e. We hope that it would be an intractable problem for the opponent to achieve any other result. There are various brute-force searches that the opponent could try, but each such search, for 50 generators, seems to involve more that 10^9 possibilities.

6. CONCLUSIONS.

We have made a case for basing cryptosystems on problems harder than NP-complete. As an illustration, we have used the undecidable word problem for groups to design a public-key cryptosystem. Public encryption is straightforward, but trapdoor insertion requires further study. An experimental system was implemented and seems resistent to initial cryptanalytic attacks. This system has a large key size and encryption time, and an excessively large expansion factor, at least 100 to 1.

ACKNOWLEDGMENT.

Dorothy Denning pointed out Brassard's work, and Jim Anderson suggested looking at homophonic ciphers. Mark Cain gave valuable help with the programming. Whit Diffie and Gilles Brassard made comments during the presentation that are incorporated into this paper.

REFERENCES.

[Adle83] L. M. Adleman, "On breaking the iterated Merkle-Hellman public- key cryptosystem," *Advances in Cryptology: Proceedings of Crypto 82,* ed. by D. Chaum et al., Plenum, 1983, pp. 303-308.

[Aho74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, 1974.

[Blum83] M. Blum, "How to exchange (secret) keys," *ACM Transactions on Computer Systems* 1, 2 (May 1983), pp. 175-193.

[Boon59] W. W. Boone, "The word problem," *Annals of Math.* 70 (1981), pp. 207-265.

[Bras79] G. Brassard, "A note on the complexity of cryptography," *IEEE Transactions on Information Theory,* IT-25, 2 (Mar. 1979), pp. 232-233.

[Bras81] G. Brassard, "An optimally secure relativized cryptosystem," *Advances in Cryptography: A report on CRYPTO 81,* ed. by A. Gersho, ECE REPT. No. 82-04, Dept. of Elect. and Computer Eng., Univ. of Calif., Santa Barbara, pp. 54-58.

[Crow63] R. H. Crowell, and R. H. Fox, *Introduction to Knot Theory,* Blaisdell, 1963.

[Diff76] W. Diffie, and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory* IT-22, 6 (Nov. 1976), pp. 644-654.

[Gary79] M. R. Gary, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman, 1979.

[Horo78] E. Horowitz, and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press, 1978.

[Lynd77] R. C. Lyndon, and P. E. Schupp, *Combinatorial Group Theory,* Springer, 1977.

[Magn66] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations,* J. Wiley (Interscience), 1966.

[Merk78] R. C. Merkle, and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory* IT-24, 5 (Sept. 1978), pp. 525-530.

[Novi55] P. S. Novikov, "On the algorithmic unsolvability of the word problem in group theory," *Trudy Mat. Inst. Steklov* 44, 143 (1955).

[Ong84] H. Ong, C. P. Schnorr, and A. Shamir, "An efficient signature scheme based on quadratic equations," *Proc. of the Sixteenth Annual ACM Symposium of Theory of Computing,* ACM 1984, pp. 208-216.

[Rabi58] M. O. Rabin, "Recursive unsolvability of group theoretic problems," *Annals of Math.* 67 (1958), pp. 172-194.

[Rabi79] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Technical Report No. TR-212, MIT Lab. for Computer Science (Jan. 1979).

[Rive78] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* 21, 2 (Feb. 1978), pp. 120-126.

[Rive79] R. L. Rivest, "Critical remarks on 'Critical remarks on some public-key cryptosystems' ," *BIT* 19 (1979), pp. 274-275.

[Rive83] R. L. Rivest, and A. T. Sherman, "Randomized encryption techniques," *Advances in Cryptology: Proceedings of Crypto 82,* ed. by D. Chaum et. al., Plenum, 1983, pp. 145-163.

[Rotm73] J. J. Rotman, *Theory of Groups: An Introduction,* Second Edition, Allyn and Bacon, 1973.

[Sha83a] A. Shamir, "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," *Advances in Cryptology: Proceedings of Crypto 82,* ed. by D. Chaum et al., Plenum, 1983, pp. 279-288.

[Sha83b] A. Shamir, "The strongest knapsack-based cryptosystem?" (presentation at Crypto 82).

[Sha83c] A. Shamir, "On the generation of cryptographically strong pseudorandom sequences," *ACM Transactions on Computer Systems* 1, 1 (Feb. 1983), pp. 38-44.

[Tarj83] R. E. Tarjan, *Data Structures and Network Algorithms,* SIAM, 1983.

[Wagn84] N. R. Wagner, "Searching for public-key cryptosystems," *Proceedings of the 1984 Symposium on Security and Privacy,* IEEE Computer Society, pp. 91-98.

[Will80] H. C. Williams, "A modification of the RSA public-key encryption procedure," *IEEE Transactions on Information Theory,* IT-26, 6 (Nov. 1980), pp. 726-729.