J. Cryptology (1988) 1: 95–105

Journal of Cryptology
© 1988 International Association for Cryptologic Research

# A Key Distribution System Equivalent to Factoring

Kevin S. McCurley[1]
Department of Mathematics, University of Southern California,
Los Angeles, CA 90089-1113, U.S.A.

**Abstract.** We propose a variation of the Diffie and Hellman key distribution scheme for which we can prove that decryption of a single key requires the ability to factor a number that is the product of two large primes. The practical advantage of such a scheme is that it will still be secure if the cryptanalyst knows a very fast algorithm for either factoring or computing discrete logarithms, but not for both. Using these keys in the ElGamal public-key cryptosystem provides a scheme for which the decryption of a message requires the ability to factor the modulus and break the original Diffie and Hellman scheme.

**Key words.** Key distribution, Discrete logarithms, Integer factoring.

## 1. The Diffie and Hellman Key Distribution Scheme

In a landmark paper in 1976, Diffie and Hellman [DH] proposed an implementation of a public key distribution scheme whose security depends on the difficulty of solving a problem in computational number theory. The Diffie and Hellman scheme may be generalized to work over an arbitrary group (or rather, a finite cyclic subgroup generated by some known element). Let $g$ be an element of the group $G$, and assume that both $g$ and an efficient algorithm for multiplying elements in $G$ are publicly known. Two users A and B who wish to agree on a common key (corresponding to a group element in some natural way) proceed as follows:

- A chooses a random number $x$, computes $g^x$, and sends the result to B over any public channel, keeping $x$ secret.
- B chooses a random number $y$, computes $g^y$, and sends the result to A over any public channel, keeping $y$ secret.
- Both A and B then use the key $g^{xy}$, which A computes as $(g^y)^x$, and B computes as $(g^x)^y$.

The original proposal of Diffie and Hellman used $G = \mathrm{GF}(p)^*$, the multiplicative group of units modulo a prime $p$, where $p$ is large ($> 10^{100}$), and $g$ a primitive root modulo $p$. The actual construction of $p$ and $g$ is easily carried out, as is the arithmetic

[1] Research supported in part by grants from the USC Faculty Research and Innovation Fund and the National Security Agency.

in the group. At present the best-known approaches to breaking the Diffie and Hellman scheme use algorithms for solving the discrete logarithm problem, which can be stated as the following:

*Given group elements $g$ and $y$, find an integer $x$ with $g^x = y$, provided one exists.*

It is conjectured that the problem of recovering the secret key $g^{xy} \pmod{p}$ for the original Diffie and Hellman scheme is equivalent to the discrete logarithm problem in $\mathrm{GF}(p)^*$, but this remains open. The best discrete logarithm algorithm for an arbitrary finite group when $g$ has known order $m$ requires $O(q^{1/2})$ group operations in the worst case, where $q$ is the largest prime factor of $m$ [PH], [Kn, pp. 9, 575–576], [P]. Faster algorithms are known for the groups $\mathrm{GF}(p)^*$ and $\mathrm{GF}(2^k)^*$; for a discussion of such algorithms, see [COS], [C], [CD], and the survey [O].

Given that the Diffie and Hellman scheme can make use of an arbitrary group, we should choose a group for which the corresponding discrete logarithm problem is difficult. There are numerous groups that we might envision using instead of $\mathrm{GF}(p)^*$ or $\mathrm{GF}(2^k)^*$, including the following:

- The group of points on an elliptic curve over a finite field (see [Ko] and [Mi]).
- The group of equivalence classes of binary quadratic forms of a given negative discriminant.
- The group $\mathrm{GL}(m, R)$ of invertible $m \times m$ matrices over a ring $R$.

It may be that the discrete logarithm problem for some of these groups is inherently much harder than others, but at present we have no way of knowing.

The Diffie and Hellman key distribution scheme is only one of many cryptosystems that have been proposed whose security depends on the presumed difficulty of solving an instance of a problem in computational number theory. Some of the schemes have turned out to be insecure, but others are still open. The most well known of these is the RSA public key cryptosystem [RSA] and its variants [W]. These schemes have the property that the success of various cryptanalytic attacks depends on the ability to factor a large integer of a special form. Since factoring is a problem that has received intense scrutiny over an extended period of time, this gives some credence to the belief that the systems are secure.

Due to the current state of computational complexity theory, there is no *proof* that any the problems mentioned so far are in fact difficult to solve, even though they have been studied by numerous researchers. The only guarantee of security that can be ascribed to these systems is, to quote from [GJ, p. 3], "I can't find an efficient algorithm, but neither can all these famous people." Given this state of affairs, a user of such a system may be more convinced of its security if breaking it requires the cryptanalyst to solve *several* problems that are presumed difficult rather than just one, and if the problems are ones that have been studied extensively (such as integer factoring).

With this in mind, we propose a variation of the Diffie and Hellman scheme working on the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$ with composite $n$ that in effect combines the security of the original scheme with the difficulty of factoring large integers, at only slightly greater expense in key construction. We prove that if the keys for these cryptosystems are chosen carefully, then any algorithm that will break the system for the published keys can be used to factor the modulus $n$ and break the original Diffie and Hellman scheme modulo the factors of $n$. Hence if the keys are chosen large enough as to render both problems intractible using current algorithms, then the system will remain secure so long as at least one of the two problems remains intractible.

The specific construction of keys proceeds as follows. We choose numbers $r$ and $s$ such that

$$\begin{gathered} 2r + 1 \text{ has a large prime factor, } 4r + 1 \text{ is prime, and } 8r + 3 \text{ is prime,} \\ s \text{ has a large prime factor, } 4s - 1 \text{ is prime, and } 8s - 1 \text{ is prime.} \end{gathered} \tag{1}$$

We set $p = 8r + 3$, $q = 8s - 1$, and $n = pq$. The size of $p$ and $q$ are dictated by the current state of the art in discrete logarithm algorithms and factoring algorithms, but the author suggests that $p$ and $q$ should have at least 100 decimal digits, and preferably 150 digits. The conditions (1) are designed for the two purposes of proving the equivalence to factoring and foiling some known attacks on factoring and computing discrete logarithms, and indeed, such keys have been suggested for use in other cryptosystems [W], [GMR], [Bl]. If so desired, the key $n$ can be constructed once and for all by a trusted center, and the factors $p$ and $q$ discarded afterward. Any two users A and B wishing to agree on a secret key use the public key $n$ in the following procedure, which we hereafter refer to as the Composite Diffie and Hellman scheme, or CDH for short:

- A chooses a random number $x$, computes $16^x \pmod{n}$, and sends the result to B over any public channel, keeping $x$ secret.
- B chooses a random number $y$, computes $16^y \pmod{n}$, and sends the result to A over any public channel, keeping $y$ secret.
- Both A and B use the key $16^{xy} \pmod{n}$.

In Section 2 we state and prove a precise theorem concerning the security of this system from which it follows that any algorithm that will break the CDH scheme for a nonnegligible proportion of the possible inputs can be used to factor $n$. It is also easy to see that any algorithm that will break the CDH scheme for a given modulus $n$ can also be used to break the original Diffie and Hellman scheme for the prime moduli that are factors of $n$. Hence the keys proposed here essentially combine the difficulty of the two problems. If the cryptanalyst discovers an algorithm that will factor $n$, then he or she is still left with the problem of breaking the Diffie and Hellman schemes for the two prime factors $p$ and $q$. Our theorem also guarantees that the cryptanalyst will be unsuccessful in discovering an algorithm for breaking the CDH scheme without factoring $n$.

The idea of using a composite modulus in the Diffie and Hellman scheme was arrived at independently by Shmuely [S] and the author [Mc], and indeed this paper may be regarded as a technical refinement of the ideas in Shmuely's paper. Shmuely proved a result that says roughly: "any algorithm that will break a Diffie and Hellman scheme with composite modulus for a nonnegligible proportion of bases $g$ can be used to factor the modulus." The algorithm for factoring $n$ uses the construction of random bases $g$, in the hope that one will satisfy certain properties. From a cryptographic standpoint, this result does not present as strong an argument as we might like for the security of the scheme, since the cryptanalyst is unlikely to be in a position to specify a random base $g$ of his or her choosing when mounting an attack. The specific choice of keys described here allows us to prove that any algorithm which will break the scheme using the published base $g = 16$ can be used to factor the modulus. This provides a more direct relationship between the problems of factoring and breaking the key distribution scheme.

## 2. Security of the CDH Scheme

Throughout this paper we use the following notation:

- $\mathbf{Z}_n^*$ denotes the multiplicative group of units in $\mathbf{Z}/n\mathbf{Z}$.
- $\langle g \rangle_n$ denotes the cyclic subgroup of $\mathbf{Z}_n^*$ generated by $g$.
- $\operatorname{ord}_n g$ denotes the order of the subgroup $\langle g \rangle_n$.
- $\# S$ denotes the cardinality of the set $S$.
- $Q(n)$ denotes the set $\{a | a \in \mathbf{Z}_n^*, \operatorname{ord}_n g \text{ is odd}\}$.
- $G(n)$ is the number of bit operations required to compute greatest common divisors of positive integers not exceeding $n$ (note that $G(n) = O(\log^2 n)$).

The problem of breaking the Diffie and Hellman scheme with a modulus $n$ and base $g$ is equivalent to the problem of computing a value of the function

$$DH(g, n, a, b)\colon \langle g \rangle_n \times \langle g \rangle_n \to \langle g \rangle_n,$$

defined by

$$DH(g, n, g^x, g^y) = g^{xy} \bmod n.$$

It is easily verifed that $DH$ is in fact a well-defined function.

We now state (in slightly different notation) a result of Shmuely connecting the problem of computing $DH$ with the problem of factoring $n$.

**Theorem** (Shmuely). *Let $0 < \delta < 1, 0 < \beta < 1, n = pq, p - 1 = 2^k p', q - 1 = 2^l q'$, with $p'$ and $q'$ odd, and $p$, $q$ odd primes. Let $A$ be an algorithm that will output $DH(g, n, a, b)$ in $T(n)$ bit operations for at least $\delta \# Q(n)$ of the inputs $g \in Q(N)$, and for each of these $g$'s, at least $\beta(\operatorname{ord}_n g)^2$ of the input pairs $(a, b) \in \langle g \rangle_n \times \langle g \rangle_n$. Then $A$ can be used to construct an algorithm that will output the factors of $n$ with probability at least 0.5 in $O(2^{k+l}\delta^{-1}\beta^{-1}(T(n) + G(n)))$ bit operations.*

Roughly speaking, this theorem may be paraphrased as saying that if we can quickly compute $DH(g, n, a, b)$ for a nonnegligible proportion of bases $g$ having odd order, then we can quickly factor $n$. The algorithm described by Shmuely for factoring $n$ involves choosing random numbers $c$, $r_1$, and $r_2$, and computing $DH(c^4, n, c^{2r_1}, c^{2r_2})$. If $c^2 \in Q(n)$, then we can prove that $c^{r_1 r_2}$ and $J = DH(c^4, n, c^{2r_1}, c^{2r_2})$ are random square roots of $c^{2r_1 r_2} \pmod n$. The probability is 0.5 that $J \not\equiv \pm c^{2r_1 r_2} \pmod n$, so that we have a good chance of factoring $n$ by computing $\gcd(c^{r_1 r_2} + J \bmod n, n)$.

The randomness of the base $g$ was a key ingredient in Shmuely's argument, since it was required to guarantee that the square roots generated by the algorithm were indeed random. As it turns out, we can show that if the keys $p$, $q$, are chosen as in (1), then the ability to compute $DH(g, n, a, b)$ for the fixed base $g = 16$ implies the ability to factor $n$.

**Theorem.** *Let $n = pq$ be as specified in* (1) *and $0 < \delta < 1$. Let $A_\delta$ be an algorithm using at most $R(n)$ bit operations that will output $DH(16, n, a, b)$ for at least $\delta(\operatorname{ord}_n 16)^2$ of the input pairs $(a, b) \in \langle 16\rangle_n \times \langle 16\rangle_n$. Then there exists an algorithm $B_\delta$ that will output p and q with probability at least* 0.5 *using at most $O(\delta^{-1}(R(n) + G(n)))$ bit operations.*

The proof of this result is quite easy, and as a first step we state the following lemma.

**Lemma.** *Let p be a prime for which $p' = (p - 1)/2$ is also prime.*

1. *If $p \equiv 3 \pmod 8$, then $\operatorname{ord}_p 2 = p - 1$ and $\langle 4\rangle_p = \langle 16\rangle_p$.*
2. *If $p \equiv 7 \pmod 8$, then $\operatorname{ord}_p 2 = p'$, $\langle 2\rangle_p = \langle 4\rangle_p = \langle 16\rangle_p$, and $-1 \notin \langle 2\rangle_p$.*

**Proof.** Clearly we must have either $\operatorname{ord}_p 2 = p'$ or $\operatorname{ord}_p 2 = 2p'$, since $\operatorname{ord}_p 2$ is a divisor of $p - 1$. It follows from Euler's criterion that

$$2^{p'} \equiv \left(\frac{2}{p}\right) \equiv (-1)^{(p^2-1)/8} \pmod p. \tag{2}$$

Assume first that $p \equiv 7 \pmod 8$. Then $\operatorname{ord}_p 2 = p'$, and since $\langle 4\rangle_p$ and $\langle 16\rangle_p$ are nontrivial subgroups of $\langle 2\rangle_p$, we have $\langle 2\rangle_p = \langle 4\rangle_p = \langle 16\rangle_p$. It also follows from (2) that 2 is a quadratic residue modulo $p$, so that $\langle 2\rangle_p$ contains only quadratic residues. Hence $-1 \notin \langle 2\rangle_p$.

If $p \equiv 3 \pmod 8$, then (2) implies that $\operatorname{ord}_p 2 = p - 1$. The subgroup $\langle 4\rangle_p$ then has order $p'$, and by the same argument as before we have $\langle 4\rangle_p = \langle 16\rangle_p$. □

We now prove our main theorem. Let $0 < \delta < 1$, $n = pq$ be as specified in (1), and let $A_\delta$ be a Monte Carlo algorithm that attempts to compute $DH(16, n, z, w)$. We now describe an algorithm $B_\delta$ for computing $p$ and $q$.

```
Algorithm $B_\delta$. (Input $n$ satisfying (1), output factors $p$ and $q$ of $n$.)
  Begin For $i = 1, 2, \ldots, 1 + \lceil 1/-\log_2(1 - \delta)\rceil$, do steps B1–B3
    B1  Choose random odd numbers $x$ and $y$ between 0 and $n$.
    B2  Execute $A_\delta$ to attempt to compute $DH(16, n, 4^x, 4^y)$.
    B3  If $A_\delta$ returns an output $D$, then compute $v = \gcd(n, D + 2^{xy} \bmod n)$. If
        $1 < v < n$, then return $v$ and $n/v$ as the prime factors of $n$.
  End
```

Clearly this algorithm has running time that is $O(\delta^{-1}(R(n) + G(n)))$ bit operations. In order to prove the main theorem, let us first observe that if $A_\delta$ produces a correct output $D = DH(16, n, 4^x, 4^y)$, then $B_\delta$ will output the factors $p$ and $q$. For this it suffices to prove that

$$D^2 \equiv 4^{xy} \pmod{n}, \tag{3}$$

$$D \not\equiv \pm 2^{xy} \pmod{n}. \tag{4}$$

By the lemma,

$$\operatorname{ord}_n 2 = \operatorname{lcm}(\operatorname{ord}_p 2, \operatorname{ord}_q 2) = \operatorname{lcm}(p-1, (q-1)/2) = 2k,$$

where $k = (4r + 1)(4s - 1)$ is odd. It follows that

$$\operatorname{ord}_n 4 = \operatorname{ord}_n 16 = k,$$

so that $16^{(k+1)/2} \equiv 4 \pmod{n}$. Hence we have

$$\begin{aligned} D &\equiv DH(16, n, 16^{x(k+1)/2}, 16^{y(k+1)/2}) \\ &\equiv 16^{xy(k+1)^2/4} \pmod{n} \\ &\equiv 2^{xyk^2+xy} \pmod{n}, \end{aligned} \tag{5}$$

so that $D^2 \equiv 2^{2xyk^2+2xy} \equiv 4^{xy} \pmod{n}$, which proves (3). It suffices now to prove (4). If we assume on the contrary that $2^{xy} \equiv D \pmod{n}$, then it follows from (5) that $2^{xyk^2} \equiv 1 \pmod{n}$. If $x$ and $y$ are odd, then this is a contradiction to the fact that $\operatorname{ord}_n 2$ is even. If on the other hand we have $2^{xy} \equiv -D \pmod{n}$, then it follows from (5) that $2^{xyk^2} \equiv -1 \pmod{n}$, and this is a contradiction to the fact that $-1 \notin \langle 2 \rangle_p$.

In order to complete the proof of the theorem, we compute the probability that $B_\delta$ will fail to output correctly the factors $p$ and $q$. Since $\langle 4 \rangle_n = \langle 16 \rangle_n$, it follows that a given element of $\langle 16 \rangle_n$ will be hit by $4^x$ and $4^y$ with probability $k^{-1}(1 + O(1/\min(p, q)))$, so that $A_\delta$ will output $DH(16, n, 4^x, 4^y)$ with probability very nearly $\delta$. Let $m = 1 + \lceil 1/-\log_2(1-\delta) \rceil$. Then

$$\begin{aligned} P(B_\delta \text{ fails}) &= P(A_\delta \text{ fails for } i = 1, \ldots, m) \\ &\approx (1-\delta)^m \\ &\leq \tfrac{1}{2}. \end{aligned}$$

This completes the proof. □

## 3. The ElGamal Schemes

In 1985 ElGamal [E] proposed new implementations of a public-key cryptosystem and digital signature scheme whose security is related to the problem of computing discrete logarithms modulo a large prime $p$. In this section we describe how to modify the schemes to work instead with a composite modulus, and discuss the relation to factoring. In the original scheme the public keys required for both systems are a large prime $p$, a primitive root $g \pmod{p}$, and, for each user, a number $y$ with $0 < y < p$. Each user generates his own public key $y$ by choosing a random value $x$ and computing $y \equiv g^x \pmod{p}$. The number $x$ is kept secret. If so desired, the keys $p$ and $g$ may be shared by all users. The ElGamal cryptosystem and signature scheme use the same keys but are otherwise different, and we therefore separate the discussion of the two.

### 3.1. *The Cryptosystem*

The ElGamal public-key cryptosystem is actually just a variation of the original Diffie and Hellman key distribution scheme. Suppose that party B wishes to send an encrypted message to party A. B consults a public file to find the keys $y$, $p$, and $g$ of the user A. In order for party B to send an encryption of the message $m$ to party A, where $1 < m < p - 1$, B first chooses a random integer $k$ with $1 < k < p - 1$, and computes $j \equiv y^k \pmod{p}$. The encrypted message is then the pair $(u, t)$, where $u \equiv g^k \pmod{p}$ and $t \equiv jm \pmod{p}$. The decryption is carried out by first computing $j \equiv u^x \pmod{p}$, and then $m \equiv tj^{-1} \pmod{p}$. This is the two-step decryption process described by ElGamal, but there is an alternative method that is slightly faster. Instead of performing two separate operations, the receiver can compute $p - 1 - x$ once when the system is set up, record it, and then compute $m$ directly by

$$m \equiv u^{p-1-x} t \pmod{p}.$$

Using this method, the decryption is essentially as fast as the corresponding operation for the RSA scheme, assuming that the keys are the same size, but the encryption still takes about twice as long as the corresponding operation in RSA. This disadvantage can be overcome if the random numbers $g^k \pmod{p}$ are precomputed in background.

Clearly, in order for the cryptanalyst to recover the secret key $x$, he must compute a discrete logarithm modulo $p$, and in order to decipher a message $m$ from the cyphertext, the cryptanalyst must solve an instance of the original Diffie and Hellman key distribution scheme, since he must recover $j$ from $u$ and $y$.

It is quite easy to modify the ElGamal schemes to work with a composite modulus. Let us assume that $n = pq$ is constructed as in (1), and let us assume that a number $n$ satisfying (1) is supplied by a trusted center, but that its prime factorization is kept secret (it is also possible for each user to have his own modulus $n$, and we shall discuss this later). Each user chooses an odd number $x$ to serve as his secret key, and computes $y \equiv 16^x \pmod{n}$. The number $y$ is published in a public file, keeping $x$ secret.

In order for someone to send an encryption of the message $m$ using the receiver's public key $y$, the sender chooses a random integer $k$ with $1 < k < n$ and computes $t \equiv my^k \pmod{n}$, and then computes $u \equiv 16^k \pmod{n}$. The encrypted message is then the pair $(u, t)$. The secret key $x$ allows the decryption to be carried out by computing

$$m \equiv t(u^x)^{-1} \pmod{n}.$$

The reason for choosing the exponent $x$ to be odd is that it is then almost certainly true that we will have $\gcd(x, \varphi(n)) = 1$. Without this it may turn out that only a few values of $t$ will appear as cyphertext. As in the original ElGamal scheme, a different value of $k$ should be used each time, for otherwise knowledge of the plaintext for a single block of the message enables the cryptanalyst to decypher other blocks of the message that are encrypted using the same $k$.

It follows from our main theorem that any algorithm that will decrypt a nonnegligible proportion of messages for a nonnegligible proportion of public keys $y$ can be used to factor the modulus $n$. In addition, it was proved by Bach [Ba] that if we have an algorithm to solve the generic instance of the discrete logarithm problem modulo $n$, then we can also use the algorithm to factor $n$ and compute discrete logarithms modulo the prime factors of $n$. Hence recovery of the secret key $x$ is in some sense equivalent to factoring the modulus. Bach's argument used the discrete logarithm problem with a variable base (similar to Shmuely), but we can give a variant of the argument as follows: assume that $n$ satisfies (1). Choose a random integer $z$ with $1 < z < n$, and compute $y \equiv 16^z \pmod{n}$. Then use the discrete logarithm algorithm to compute an integer $x$ with $y \equiv 16^x \pmod{n}$. There at least four distinct choices of $x$ with $1 < x < n$, and the probability is at least $\frac{1}{2}$ that $z - w$ will be a nonzero multiple of $k = \operatorname{ord}_n 16$. From $k$ it is a simple matter to recover the prime factors of $n$. A slight modification of the ideas of Miller [M] can be used to give a similar argument for the case of a general odd modulus $n$, provided the base $g$ has $\operatorname{ord}_n g$ divisible by all of the large primes dividing $\varphi(n)$. For the general modulus however, we lose the close connection between decryption and factoring.

Instead of having everybody using the same modulus $n$, it is also possible for every user to use a different $n$. In this case the public file will be larger, but the decryption process can be speeded up if the exponent $\varphi(n) - x$ is computed once and for all when the system is set up. The decryption process then uses

$$m \equiv tu^{\varphi(n)-x} \pmod{n},$$

eliminating the need for the Euclidean algorithm.

## 3.2. *The Signature Scheme*

The original ElGamal signature scheme uses the same keys as the cryptosystem. In order for a user to sign the message $m$, he chooses a random number $k$ with $0 < k < p - 1$ and $\gcd(k, p - 1) = 1$, and computes $r \equiv g^k \pmod{p}$. He then computes $s \equiv k^{-1}(m - xr) \pmod{p - 1}$, using his secret key $x$. The signature of the message $m$ consists of the pair $(r, s)$. Verification of the signature is accomplished easily since the receiver simply uses the public key $y$ and verifies that the congruence $g^m \equiv y^r r^s \pmod{p}$ is satisfied. It is presumed that the problem of finding such an $r$ and $s$ is computationally infeasible if $x$ is unknown. It is unknown whether the success of a forgery attempt is equivalent to computing a discrete logarithm or breaking a Diffie and Hellman scheme.

In the original ElGamal schemes, it was possible for everyone to use the same prime $p$, and indeed this is still possible in the cryptosystem using a composite modulus as described in Section 3.1. In order to modify the signature scheme to use a composite modulus, each user must have a different public key $n$. In order for someone to sign messages in the composite scheme, he must know $\varphi(n)$, and in this case he can factor $n$ (see [RSA]). If an unscrupulous user of the system knows the factorization of another user's public $n$, then, if he can also quickly compute discrete logarithms modulo the factors, he can forge other users's signatures and read the other user's encrypted messages. The whole point of the composite scheme is to use the difficulty of both factoring and computing discrete logarithms to provide added security. With this understanding, the signature scheme of ElGamal requires only a slight modification in order to use a composite modulus. In order to sign the message $m$, the user chooses a random number $k$ with $\gcd(k, \varphi(n)) = 1$ and computes $r \equiv 16^k \pmod{n}$. He then computes $s$ with $s \equiv k^{-1}(m - xr) \pmod{\varphi(n)}$. As before, the signature is the pair $(r, s)$, and the signature is assumed to be authentic if $16^m \equiv y^r r^s \pmod{n}$.

There is very little that can be said concerning the security against forgery for either the original ElGamal signature scheme or the ElGamal signature scheme with composite modulus, since there is no known argument to show that the ability to carry out a key or forgery attack implies the ability to solve one of the computational problems mentioned so far. It appears at first sight that the problem of recovering the secret key is equivalent to computing a discrete logarithm modulo $p$, but in practice the cryptanalyst would also have access to signatures of some messages, which is more information than was available in the discrete logarithm problem. It is not clear how the cryptanalyst can gain any advantage from this information.

## 4. Further Remarks on Key Selection

The specific choice of keys that are described here are by no means the only ones for which it is possible to prove a direct link between the Diffie and Hellman scheme and factoring. The choice of $n$ described in (1) is simply convenient for exhibiting a number $g$ whose orders modulo the prime factors of $n$ have certain desired properties. The base $g = 16$ may however have some advantage for implementation due to its simple binary representation.

We might wonder if it is easy to construct the numbers $r$ and $s$. Let $m$ be an integer for which $2m + 1$, $12m + 5$, and $24m + 11$ are prime. If we set $r = 3m + 1$, then $r$ will satisfy (1). Let $\pi_1(x)$ denote the number of such integers $m$ between 1 and $x$. It is a consequence of a well-known conjecture in analytic number theory (see [BH]) that

$$\pi_1(x) \sim \frac{Cx}{\log^3 x}$$

as $x \to \infty$, where

$$C = 18 \prod_{p \geq 5} \left(1 + \frac{1 - 3p}{(p-1)^3}\right) \approx 11.4.$$

As a consequence we should expect that a randomly chosen number $m$ near $10^{100}$ will satisfy the required conditions with probability at least $9.3 \times 10^{-7}$. Hence we should expect to find such an integer $m$ before we examine a million or so consecutive numbers. In fact, a simple sieve procedure similar to the sieve of Eratosthenes will speed up the search for keys considerably. Note that the number of such integers $m$ with 100 decimal digits probably exceeds $10^{93}$, so that an exhaustive search for the factor $p$ is hopeless. We may also formulate a similar argument for the number of integers $k$ such that $2k + 1$, $24k + 11$, and $48k + 23$ are prime, and if we set $s = 6k + 3$, then we have numerous examples satisfying (1).

If $p$ and $q$ had been chosen so that $p - 1$ or $q - 1$ had no large prime factors, then the scheme would have been vulnerable to an attack based on factoring the modulus $n$ using the Pollard $p - 1$ method followed by the Pohlig–Hellman algorithm for computing discrete logarithms. The condition (1) is furthermore designed to thwart an attempt to factor $n$ via the Williams $p + 1$ method.

It should also be observed that the construction of the primes $p$ and $q$ satisfying (1) is facilitated by the fact that if $m > 3$ is prime, then $p = 2m + 1$ is prime if and only if $3^m \equiv 1 \pmod{p}$.

## References

[Ba] E. Bach, Discrete Logarithms and Factoring, Technical Report UCB/CSD 84/186, Computer Science Division (EECS), University of California, Berkeley, California, June, 1984.

[BH] P. T. Bateman and R. A. Horn, A heuristic asymptotic formula concerning the distribution of prime numbers, *Mathematics of Computation*, **16** (1962), 363–367.

[Bl] M. Blum, Coin flipping by telephone, *Proc. IEEE Spring COMPCON*, 1982, pp. 133–137.

[C] D. Coppersmith, Fast evaluation of discrete logarithms in fields of characteristic 2, *IEEE Transactions on Information Theory*, **30** (1984), 587–594.

[CD] D. Coppersmith and J. H. Davenport, An application of factoring, *Journal of Symbolic Computation*, **1** (1985), 241–243.

[COS] D. Coppersmith, A. Odlyzko, and R. Schroeppel, Discrete logarithms in GF($p$), *Algorithmica*, **1** (1986), 1–15.

[DH] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, **22** (1976), 472–492.

[E] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, **31** (1985), 469–472.

[GJ] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[GMR] S. Goldwasser, S. Micali, and R. Rivest, A digital signature scheme secure against adaptive chosen message attack (extended abstract), *Discrete Algorithms and Complexity; Proceedings of the Japan–U.S. Joint Seminar*, Academic Press, Orlando, FL, 1987.

[Kn] D. E. Knuth, *The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, MA, 1973.

[Ko] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation*, **48** (1987).

[L] H. W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics*, to appear.

[Mc] K. S. McCurley, Digital Signatures and Public-key Cryptosystems Based on Discrete Logarithms and Factoring, Unpublished manuscript, 1986.

[M] G. Miller, Riemann's hypothesis and tests for primality, *Journal of Computer and System Science*, **13** (1976), 300–317.

[Mi] V. Miller, Use of elliptic curves in cryptography, *Advances in Cryptology* (Proceedings of Crypto '85), Lecture Notes in Computer Science, Vol. 218, Springer-Verlag, New York, 1986, pp. 417–426.

[O] A. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, *Advances in Cryptology* (Proceedings of Eurocrypt 84), Lecture Notes in Computer Science, Vol. 209, Springer-Verlag, New York, 1985, pp. 224–314.

[PH] S. Pohlig and M. Hellman, An improved algorithm for computing discrete logarithms over GF($p$) and its cryptographic significance, *IEEE Transactions on Information Theory*, **24** (1978), 106–110.

[P] J. M. Pollard, Monte Carlo methods for index computation (mod $p$), *Mathematics of Computation*, **32** (1978), 918–924.

[RSA] R. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, *Communications of the ACM*, **21** (1978), 120–126.

[S] Z. Shmuely, Composite Diffie–Hellman Public-Key Generating Systems Are Hard To Break, Technical Report No. 356, Computer Science Department, Technion-Israel Institute of Technology, February, 1985.

[W] H. C. Williams, A modification of the RSA public key encryption system, *IEEE Transactions on Information Theory*, **26** (1979), 726–729.