Study of RSA and Proposed Variant against Wiener's Attack

Justin Jose
T.E Computers, Fr.CRIT, VASHI
Fr. Agnel Boys Hostel Sec 9-A Vashi, Navi Mumbai-400703

Siddharth Raina
T.E Computers, Fr.CRIT, VASHI
Fr. Agnel Boys Hostel Sec 9-A Vashi, Navi Mumbai-400703

Sushant Pawar
T.E Computers, Fr.CRIT, VASHI
C-22 B.E.S.T Qtr, Parel Mumbai12

Shriket Pai
T.E Computers, Fr.CRIT, VASHI
B-11 Kailash Vihar, Ghatkopar(W) Mumbai-86

ABSTRACT

The paper discusses encryption schemes such as public key algorithms (RSA) and One Time Pads. It also discusses various attacks on the RSA algorithm. A brief introduction to Modular Arithmetic, which is the core arithmetic of almost all public key algorithms, has been given. In this paper we propose a variant to the RSA algorithm which is effective against Wiener's Short Secret Exponent attack. The security and the efficiency of the proposed variant have also been discussed.

General Terms

Public key, private key, X-or, encryption, cryptanalysis, cryptosystem, efficiency,

Keywords

RSA, one-time pad, Wiener's attack, modular arithmetic, plaintext, ciphertext.

1.INTRODUCTION

The data transferred from one system to another over public network can be protected by the method of encryption. On encryption the data is encrypted/scrambled by any encryption algorithm using the 'key'. Only the user having the access to the same 'key' can decrypt/de-scramble the encrypted data. This method is known as private key or symmetric key cryptography. There are several standard symmetric key algorithms defined. Examples are AES, 3DES etc. These standard symmetric algorithms defined are proven to be highly secured and time tested. But the problem with these algorithms is the key exchange. The communicating parties require a shared secret, 'key', to be exchanged between them to have a secured communication. The security of the symmetric key algorithm depends on the secrecy of the key. Keys are typically hundreds of bits in length, depending on the algorithm used. Since there may be number of intermediate points between the communicating parties through which the data passes, these keys cannot exchange online in a secured manner. In a large network, where there are hundreds of system connected, offline key exchange seems too difficult and even unrealistic. This is where public key cryptography comes to help. Using public key algorithm a shared secret can be established online between communicating parties with out the need for exchanging any secret data.

In public key cryptography each user or the device taking part in the communication have a pair of keys, a public key and a private key, and a set of operations associated with the keys to do the cryptographic operations. Only the particular user/device knows the private key whereas the public key is distributed to all users/devices taking part in the communication. Since the knowledge of public key does not compromise the security of the algorithms, it can be easily exchanged online.

In public key cryptography, keys and messages are expressed numerically and the operations are expressed mathematically. The private and public key of a device is related by the mathematical function called the one-way function. One-way functions are mathematical functions in which the forward operation can be done easily but the reverse operation is so difficult that it is practically impossible. In public key cryptography the public key is calculated using private key on the forward operation of the one-way function. Obtaining of private key from the public key is a reverse operation. If the reverse operation can be done easily, that is if the private key is obtained from the public key and other public data, then the public key algorithm for the particular key is cracked. The reverse operation gets difficult as the key size increases. The public key algorithms operate on sufficiently large numbers to make the reverse operation practically impossible and thus make the system secure. For e.g. RSA algorithm operates on large numbers of thousands of bits long.

2.MATHEMATICAL BACKGROUND – MODULAR ARITHMETIC

Modular Arithmetic [1] is also known as "clock" arithmetic. Basically $a \equiv b \pmod{n}$ if $a= b+kn$ for some integer k. If a is non negative and b is between 0 and n, one can think of b as the remainder of a when divided by n. Sometimes, b is called the residue of a, modulo n. a is called congruent to b, modulo n. '≡' denotes congruence.

The set of integers from 0 to n-1 form what is called a complete set of residues modulo n. This means that, for every integer a, its residue modulo n is some number from 0 to n-1. This operation is called modular reduction.

The general problem that arises during public key encryption schemes is to find two number 'x' such that 1 = (a*x) mod n where 'a' is the one of the keys used in public key encryption. This is also written as $a^{-1} \equiv x \pmod{n}$. This modular inverse problem is difficult to solve. Sometimes it has a solution sometimes not. For example inverse of 5 modulo 14 is 3, and on the other hand 2 has no inverse modulo 14.

In general $a^{-1} \equiv x \pmod{n}$ has a unique solution if a and n are relatively prime. If n is a prime number then every number from 1 to (n-1) is relatively prime to n and has exactly one inverse modulo n in that range.

3. ENCRYPTION TECHNIQUES

The various encryption techniques on which this paper is based are

1. One-Time pads[1]
2. Public key algorithm(RSA)

3.1One-Time Pads

This encryption scheme was invented in 1917 by Major Joseph Mauborgne and AT & T's Gilbert Vernam[2]. Classically, a one-time pad is nothing more than a large non repeating set of truly random key letters, written on sheets of paper, and glued together in a pad. In its original form, it was a one-time tape for teletypewriters. The sender uses each key letter on the pad to encrypt exactly one plain text character. Encryption is the *Addition Modulo 26* of the plaintext character and the one-time pad key character. Each key letter is used exactly once, for only one message. The sender encrypts the message and then destroys the used pages of the pad or used section of the tape. The receiver has an identical pad and uses each key on the pad, in turn, to decrypt each letter of the ciphertext. The receiver destroys the same pad pages or tape sections after decrypting the message. New message-new key letters.

eg. If the message is :

COPYLEFT

And the key sequence from the pad is :

EFWJMDSZ

Then the ciphertext is:

HUMIYIYT.

Assuming an eavesdropper can't get access to the one-time pad used to encrypt the message this scheme is perfectly secure. A given ciphertext message is equally likely to correspond to any possible plaintext message of equal size.

Since every key sequence is equally likely, an adversary has no information with which to cryptanalyze the ciphertext. The key sequence could just as be:

DTIVSDBT

This would decrypt to:

LOVERMAN

This point bears repeating: Since every plaintext message is equally possible, there is no way for the cryptanalyst to determine which plaintext message is the correct one. A random key sequence added to a nonrandom plaintext message produces completely random ciphertext message and no amount of computing power can change that.

The caveat is that the key letters have to be generated randomly. Any attack this scheme will be against the method used to generate the key letters. The other important point is that one can never use the same key sequence again.

The idea of one-time pad can be easily extended to binary data. Instead of one-time pad consisting of letters, use a one-time pad of bits.

Since the key bits must be random and can never be used again, the length of the key sequence must be equal to the length of the message. Practically this encryption scheme works best for short messages.

3.2Public Key Encryption

Encryption is a process in which the sender encrypts/scrambles the message in such a way that only the recipient will be able to decrypt/ descramble the message.

Consider a device B whose private key and public key are P_B and U_B respectively. Since U_B is public key all devices will be able to get it. For any device that needs to send the message 'Msg' in a secured way to device B, it will encrypt the data using B's public key to obtain the cipher text 'Ctx'. The encrypted message, cipher text, can only be decrypted using B's private key. On receiving the message the B decrypts it using its private key P_B Since only B knows its private key P_B, none other including A can decrypt the message.

Ctx = Encrypt (Msg , U_B)

Msg = Decrypt (Ctx , P_B)

3.2.1RSA Algorithm

Of all the public-key algorithms proposed over many years, RSA[3] is by far the easiest to understand and implement. It is also the most popular. Named after the three inventers – Ron Rivest, Adi Shamir, and Leonard Adleman – it has since withstood years of extensive cryptanalysis. Although the cryptanalysis neither proved nor disproved RSA's security, it does suggest a confidence level in the algorithm.

RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits or even larger) prime numbers. Recovering the plain text from the public key and the ciphertext is conjectured to be equivalent to factoring the product of the two primes.

To generate the two keys, choose two random large prime numbers, p and q. for maximum security, choose p and q of equal length. Compute the product:

$$n=pq$$

Then randomly choose the encryption key, e, such that e and (p-1)(q-1) are relatively prime. Finally, use the extended

Euclidean Algorithm[1] to compute the decryption key, d, such that

$ed\equiv1 \bmod (p-1)(q-1)$

In other words,

$d=e^{-1} \bmod ((p-1)(q-1))$

Note that d and n are also relatively prime. The numbers e and n are the public key; the number d is the private key. The two primes, p and q, are no longer needed. They should be discarded but, never revealed.

To encrypt a message m, first divide it into numerical blocks smaller than m. That is, if both p and q are 100-digit primes, then n will have just under 200 digits and each message blocks, m_i, should be just under 200 digits long. The encrypted message, c, will be made up of similarly sized message blocks, c_i, of about the same length. The encryption formula is simply

$C_i = m_i^e \bmod n$

To decrypt a message, take each encrypted block c_i and compute

$m_i = c_i^d \bmod n$

eg: In this example we will group the characters into blocks of three and compute a message representative integer for each block.

ATTACKxATxSEVEN = ATT ACK XAT XSE VEN
In the same way that a decimal number can be represented as the sum of powers of ten, e.g.
135 = 1 x 102 + 3 x 101 + 5,
we could represent our blocks of three characters in base 26 using A=0, B=1, C=2, ..., Z=25

ATT = 0 x 262 + 19 x 261 + 19 = 513
ACK = 0 x 262 + 2 x 261 + 10 = 62
XAT = 23 x 262 + 0 x 261 + 19 = 15567
XSE = 23 x 262 + 18 x 261 + 4 = 16020
VEN = 21 x 262 + 4 x 261 + 13 = 14313

In this system of encoding, the maximum value of a group (ZZZ) would be 263-1 = 17575, so we require a modulus n greater than this value.

1. We "generate" primes p=137 and q=131.

2. n = pq = 137.131 = 17947
   φ = (p-1)(q-1) = 136.130 = 17680

3. Select e = 3
   check gcd(e, p-1) = gcd(3, 136) = 1, OK and
   check gcd(e, q-1) = gcd(3, 130) = 1, OK.

4. Compute d = e-1 mod φ = 3-1 mod 17680 = 11787.

5. Hence public key, (n, e) = (17947, 3) and private key (n, d) = (17947, 11787).

To encrypt the first integer that represents "ATT", we have
$c = m^e \bmod n$ = 5133 mod 17947 = 8363.
We can verify that our private key is valid by decrypting
$m' = c^d \bmod n$ = 836311787 mod 17947 = 513.

Overall, our plaintext is represented by the set of integers m

(513, 62, 15567, 16020, 14313)

We compute corresponding ciphertext integers $c = m^e \bmod n$, (which is still possible by using a calculator)

(8363, 5017, 11884, 9546, 13366)

You are welcome to compute the inverse of these ciphertext integers using $m = c^d \bmod n$ to verify that the RSA algorithm still holds. However, this is now outside the realms of hand calculations.

3.2.2Previous Attacks on RSA

In this section we summarize several previously known attacks on the RSA public-key cryptosystem relevant to this work. We follow the presentation of the recent survey

3.2.2.1Factoring

The most straight forward attack on RSA is factorization of the modulus n = pq. Once a factor p is discovered, the factor q = n/p may be computed, so φ(n) = n − p − q + 1 is revealed. This is enough to compute d ≡ e−1 mod φ(n).

The current fastest method for factoring is the General Number Field Sieve [4]. It has a running time of exp $(c + o(1)) \cdot (\log N)^{1/3} (\log \log N)^{2/3}$ for some $1 < c < 2$. The size of N is chosen to foil this attack. The largest integer that has been successfully factored using this method was the 512-bit RSA challenge modulus RSA-155, factored in 1999 using a massive distributed implementation of GNFS on the Internet [6]. Even though the speed of computer hardware continues to accelerate, it seems unlikely that the best factoring algorithms will be able to factor say 1024-bit RSA modulo in the next twenty years.

3.2.2.2 Hastad's Attack on Broadcasted Messages

In order to speed up RSA encryption (and signature verification) it is useful to use small value for the public exponent e, say e = 3. However, this opens up RSA to the following attack, discovered by Hastad [5].

Let us start with a simpler version. Suppose Bob wishes to send the same message M to k recipients, all of whom are using public exponent equal to 3. He obtains the public keys N_i , e_i for i = 1, . . . , k, where $e_i = 3$ for all i. Naively, Bob computes the ciphertext $C_i = M^3 \bmod N_i$ for all i and sends C_i to the ith recipient.

A simple argument shows that as soon as k ≥ 3, the message M is no longer secure. Suppose Eve intercepts C_1 , C_2 , and C_3 , where $C_i = M^3 \bmod N_i$. We may assume $\gcd(N_i , N_j) = 1$ for all $i \neq j$ (otherwise, it is possible to compute a factor of one of the N_i 's.) By the Chinese Remainder Theorem, she may compute $C \in Z^*_{N1N2N3}$ such that $C \equiv C_i \bmod N_i$. Then $C \equiv M^3 \bmod N_1 N_2 N_3$; however, since $M < N_i$ for all i, we have $M^3 < N_1N_2N_3$. Thus $C = M^3$ holds over the integers, and Eve can compute the cube root of C to obtain M.

Hastad proves a much stronger result. To understand it, consider the following naive defense against the above attack. Suppose Bob applies a pad to the message M prior to encrypting it so that the recipients receive slightly different messages. For instance, if M is m bits long, Bob might encrypt i · 2m + M and send this to the ith recipient. Hastad proved that this linear padding scheme is not secure. In fact he showed that any fixed polynomial applied to the message will result in an insecure scheme.

3.2.2.3Coppersmith Attack on Short Random Pads

Like the previous attack, this attack exploits a weakness of RSA with public exponent $e = 3$. Coppersmith showed that if randomized padding is used improperly then RSA encryption is not secure [6]. Coppersmith addressed the following question: if randomized padding is used with RSA, how many bits of randomness are needed?

To motivate this question, consider the following attack. Suppose Bob sends a message M to Alice using a small random pad before encrypting. Eve obtains this and disrupts the transmission, prompting Bob to resend the message with a new random pad. The following attack shows that even though Eve does not know the random pads being used, she can still recover the message M if the random pads are too short.

For simplicity, we will assume the padding is placed in the least significant bits, so that $C_i = (2^m M + r_i)^e \bmod N$ for some small m and random $r < 2m$. Eve now knows

$$C_1 = (2^m M + r_1)^e (\text{mod } N) \text{ and}$$

$$C_2 = (2^m M + r_2)^e (\text{mod } N)$$

for some unknown M , r_1 , and r_2 . Define $f(x, y) := x^e - C_1$ and $g(x, y) := (x+y)^e - C_2$. We see that when $x = 2^m M + r_1$, both of these polynomials have $y = r_2 - r_1$ as a root modulo N . We may compute the resultant $h(y) := \text{Resx}(f, g)$ which will be of degree at most e^2 Then $y = r_2 - r_1$ is a root of h(y) modulo N . If $|r_i| < (1/2)N^{(1/e)2}$ for i = 1, 2 then we have that $|r_2 - r_1| < N^{1/e}$. By Coppersmith's Theorem we may compute all of the roots h(y), which will include $r_2 - r_1$. Once $r_2 - r_1$ is discovered, we may use a result of Franklin and Reiter [9] to extract M

THEORM (Univariate Coppersmith): *Let a monic polynomial f(x) of degree d with integer coefficients and integers X, M be given. Suppose $X < M^{(1/d)-Є}$ for some $Є > 0$. There is an algorithm to find all $x_0 Є Z$ satisfying $|x| < X$ and $f(x_0) \equiv 0 \bmod M$. This algorithm runs in time $O(T_{LLL}(md, m \log M))$ where $m = O(k/d)$ for $k = \min\{1/\epsilon \log M\}$.*

3.2.2.4 Wiener's Attack on Short Secret Exponent

To speed up RSA decryption and signing, it is tempting to use a small secret exponent d rather a random $d \leq φ(N)$. Since modular exponentiation takes time linear in log2 d, using a d that is substantially shorter than N can improve performance by a factor of 10 or more. For instance, if N is 1024 bits in length and d is 80 bits long, this results in a factor of 12 improvement while keeping d large enough to resist exhaustive search.

Unfortunately, a classic attack by Wiener [7] shows that a sufficiently short d leads to an efficient attack on the system. His method uses approximations of continued fractions. This attack is stated in the following theorem.

THEORM: *Suppose $N = pq$ and $\sqrt{(N/2)} < q < p < \sqrt{N}$. Furthermore $d < ⅓N^{(1/4)}$. There is an algorithm which, given N and e, generates a list of length log(N) of candidates for d, one of which will equal d. This algorithm runs in time linear in log(N).*

3.2.2.5 Cryptanalysis via the Defining Equation

Since $ed \equiv 1 \bmod φ(N)$, this implies there exists an integer k such that

$$ed + k(N + 1 - (p + q)) = 1.$$

As discussed earlier, a break of the RSA public key cryptosystem can be defined in several ways. Most obviously the scheme is broken if an attacker is able to recover the secret exponent d. Since factorization of the modulus N = pq leads to recovery of the private key d, this is also a total break. All of the attacks presented in subsequent chapters are of this type, and involve either a direct computation of the private key d or one of the factors p of the public modulus N , given the public key information N, e alone.

We note that this immediately allows the recovery of the factorization of N ; indeed, when $s = p + q$, then p and q are the two roots of the equation $x2 - sx + N = 0$.

4.PROPOSED VARIENT OF RSA

As stated above in Wiener's attack on Short Secret Exponent, if the value of the private key i.e. d is upto one quarter the size of n and e < n then the attack will recover d, but on the other hand this small value of d enhances the efficiency of the cryptosystem.

Based on this we propose a new variant of RSA which uses a comparatively small value of d and is almost resistant to Wiener's attack.

ASSUMPTIONS:

1. **'a'** is the plaintext message.
2. **'b'** is the RSA ciphertext.
3. <e , n> is the public key.
4. <d> is the private key.
5. **'x'** is the randomly generated number
6. **f(b,x)** is the selected function which is to be applied on the RSA ciphertext.

ENCRYPTION ALGORITHM.

1. Compute the values of the public key and private key.
2. Encrypt 'a' using the RSA algorithm.
3. Generate the random number[1] x, such that it has many factors and is hard to predict using brute force.
4. Let β = f(b,x) be a function on b and x and should have the following properties:
    a. The function should be one-to-one onto function.
    b. The function should be invertible
    c. It should always return an integer value.
5. Encrypt x using one-time pad encryption technique, and along with the encrypted message β transmit it to the recipient.

DECRYPTION ALGORITHM

1. Decrypt x using the shared one-time pad
2. Apply inverse of function f(b , x) on β.

$b=f^{-1}(\beta, x)$

3. Retrieve original plaintext ‘a’ using private key d

$$\mathbf{a = b^{d} \bmod n}$$

This above mentioned function f(b , x) need not necessarily be an arithmetic function. It can be logical, discrete and even can be a quantum operation.

This function should always be changed with every session of communication. The functions selected for this purpose should be represented by a binary value and these bits should be padded to the random number ‘x’. When receiver receives the one-time pad message he should be able to recognize the function applied and thus the inverse of the function should be again applied on β using the recovered value x and thus calculate ‘b’. This can be achieved by keeping a log of such functions in the encryption software such that it reads the binary representation of the function and search for the same in the log and apply its inverse.

The idea of using one-time pads is that this encryption technique works best for short messages and this scheme is very hard to break as the probability of committing mistakes is very low and so is the probability of breaking this encryption technique. We can also use the Hybrid one-time pads for this purpose. The most wonderful feature of this technique is that many keys lead ultimately to a legitimate message so the attacker can never determine when he has actually broken the code.

Moreover, as we choose a comparatively small value of d, this decreases the complexity of RSA algorithm upto 1/10th of the usual one. Thus this variant of RSA is more efficient and is easy to implement too. We just need to select such functions that can be applied to the RSA code and value of x should be actually random because most of the random generators have non-random properties.

We can illustrate all our proposed theory by the following **Example:**

First consider a message ‘a’ and public key < e , n> and private key <d>.

Let a=62415, e=197, n=1363, d=85.

We take the block length as 2.Thus the message can be re-written as:

62 41 05

As we can see the digit 5 has been padded with 0.

We calculate the code ‘b’ by using:

$b=a^{e} \bmod n;$

$(62^{197}) \bmod 1363=91$

$(41^{197}) \bmod 1363=389$

$(05^{197}) \bmod 1363=701$

The encrypted code is: 91 389 701.

Now suppose the randomly generated number is 783(We have taken a small value for x. But in general x should be minimum 128 bits).we choose the function to be b^x(X-or).Actually this function should not be practically used as X-or encryption is easy to break. But we have taken this just to make calculations easy.

Thus β=852 650 434.

Now this message is transmitted along with 783 encrypted with Hybrid one-time pads in which the binary representation of the function X-or is padded.

Now the receiver will get the dual encrypted message and the value of randomly generated number through one-time pad decryption. Thus the receiver decrypts the code as below:

b=β^x(X-or).

852^783=91.

650^783=389.

434^783=701.

b=91 389 701.

Now the original message ‘a’ can be calculated as:

$a=b^{d} \bmod n.$

$(91^{85}) \bmod 1363=62$

$(389^{85}) \bmod 1363=41$

$(701^{85}) \bmod 1363=5$

Recovered a=62 41 5.

4.1 Security and Advantages

The major problem of using short private key in RSA algorithm is that the private key can be retrieved using approximations of continued fractions. But the disadvantage of using long private key is that the complexity of the algorithm highly increases. In hardware, RSA is among thousand times slower than DES. The fastest VLSI hardware implementation for RSA with 512-bit modulus has a throughput of 64 kilo bit per second[8]. There are also chips that perform 1024-bit RSA encryption. These numbers may change slightly as technology changes, but RSA will never approach the speed of symmetric algorithms.

By using the proposed variant the size of the private key ‘d’ can be kept short in turn reducing the complexity related to the large private key encryption.

The basic idea behind this variant is to dually encrypt the plaintext one RSA and other by such an algorithm which is hard to break (like ONE-TIME PADs).

The chances of a short secret exponent attack are very much less in this variant because

- If the attacker is successful in getting the private key ‘d’ using the Wiener’s attack on short secret exponent he has no knowledge of the random number x, used to calculate function f(b , x).
- Since x can return more than one legitimate answer the attacker will have many decrypted messages having no knowledge of the correct message.
- The function applied on x and b is varied constantly thus finding the inverse of the function also becomes difficult.
- Decrypting the value of x is also difficult as the one-time pad encryption on small values is hard to decrypt

attacked. Moreover, with every session the value of x and the function gets changed and x is never repeated.

Thus this variant is much more efficient than using large public and private key values.

5.ACKNOWLEDGMENTS

We are grateful to Mr.Bruce Schneier author of *Applied Cryptography* which has been of great help in studying the basic principles of Public-key Cryptography.

We are also grateful to Mr.Glenn Durfee author of the paper

Cryptanalysis of RSA using Algebraic and Lattice Methods which has been a basic reference for studying various attacks on the RSA algorithm.

6.REFERENCES

[1] Applied Cryptography by Bruce Schneier ISBN 9971-51-348-X.

[2] D. Khan, *The Code Breakers: The story of secret Writing,* New York: Macmillan publishing co., 1967.

[3] R.L. Rivest and A. Shamir, "How to Expose an Eavesdropper" *Communication of the ACM, v.27, n. 4 april 1984*.

[4] D. Gordon. Discrete Logarithms in GF(p) using the Number Field Sieve, SIAMJ. Discrete Math. , Vol.6,pp.124-138,1993

[5] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of 512 bit RSA key using the number field sieve. In proceedings Eurocrypt 2000, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000. Factorization announced in August, 1999.

[6] J. Hastad. Solving simultaneous modular equations of low degree. SIAM Journal on Computing, vol. 17, no. 2, pp. 336–341, 1988.

[7] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology, vol. 10, pp. 233–260, 1997.

[8] E.F Brickell, "Survey of Hardware Implementations of RSA," *Advances in Cryptology*-CRYPTO '89 *Proceedings,* Springer-Verlag,1990,

[9] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low exponent RSA with related messages. In proceedings Eurocrypt'96, Lecture Notes in Computer Science, vol.1070, Springer-Verlag, pp.1–9, 1996.

[10] M. Wiener. Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, vol. 36, no. 3, pp. 553–558, 1990.