Volume 4, No. 2, Jan-Feb 2013

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Efficient RSA Based Key Authenticated Prime Fibonacci Cryptosystem

Archana Raghuvamshi*
Assistant Professor
Department of Computer Science
Adikavi Nannaya University,
Rajahmundry,Andhra Pradesh
archana_raghuvamshir@yahoo.com

Prof.P.Premchand
Department of CSE
Osmania Univeristy
Hyderabad, Andhra Pradesh
University College of Engineering
p.premchand@uceou.edu

Radhika Sivalenka
Senior Programmer
Kyocera Document Solutions
Lake Hiawatha, NJ 07034
United States of America(USA)
radhika.sivalenka@da.kyocera.com

Abstract: Key Authentication is a process of gauranteeing that the public key of a client A which is held by client B does in reality belong to client A. This paper presents a novel framework for the generic construction of RSA based key Authenticated prime fibonacci crypto scheme which produces more efficient schemes than the one know before.

Keywords: Key Authentication, RSA, Prime Fibonacci Cryptosystem.

I. INTRODUTION

Key agreement or key exchange algorithm proposed by Deffie-Hellman in 1976 [1]has suffered from man-in-middle attack, because no authentication procedure is coupled with the exchanged message. In 1992,S. M. Bellovin and M. Merrit[2] has proposed a new scheme "Encrypted key exchange: password-based protocols secure against dictionary attacks" in which users are permitted to use easy-to-remember passwords. A password is shared between two parties, and these two parties may use the shared password to negotiate a common session key (secret key). Thus, two parties can communicate with each other securely.

Steiner, G. Tsudik and M. Waidner, has proposed refinement and extension of encrypted key exchange: a three-party EKE protocol (STW-3PEKE) base on EKE protocols [6] in 1995. In this protocol, each user shares an easy- to - remember password with a trusted third party, server and each user can securely exchange their session keys (secret keys) via the server. In 1999, D. Seo and P. Sweeney [3] proposed a Simple authenticated key agreement algorithm which uses a low entropy password for authenticating the two parties. Later, Y. M. Tseng [4] has pointed out the weakness involved in a simple authenticated key agreement protocol and further proposed an improved scheme to repair the security flaw in Seo and Sweeney's protocol. Overriding their claims in 2000, W. C. Ku and S. D. Wang [5] has done the cryptanalysis of modified authenticated key agreement protocol by showing the improved protocol is still vulnerable to the backward replay attack without modification and the modification attack. Under the backward replay attack, the adversary can impersonate one communicating party to fool the other one into believing the wrong session key by replaying the exchanged message. They further proposed an improved scheme to withstand those attacks. In 2003, Hsu et al. [7] and Chang et al. [8] separately pointed that Ku and Wang's[5] improved protocol is still vulnerable to suffer from modification attacks and they separately proposed the security enhancement in Ku and Wang's protocol[5].

This paper presents An Efficient RSA Based Key Authenticated Prime Fibonacci Cryptosystem. The paper is organized as follows: Section 2 describes the background of the proposed cryptosystem.In Section 3 we have listed the notation used in this novel cryptosystem. Section 4, describes about the proposed cryptosystem.Section 5 describes about the results and discussion and concluding remarks are made in Section 6.

II. BACKGROUND

When defining a cryptosystem, details must be given of : i)The alphabets *M* and *C* where M is a Message and C is a Ciphertext ii)the keyspace *K* and how keys are to be chosen iii)The encryption and decryption algorithms *f* and *g* iv)The method of blocking (if any). The security of a cryptosystem lies in the keys. If we know the keys then we can encrypt and decrypt messages.Catherine might know everything about a cryptosystem and he might be able to intercept messages.Even with all of this information, he should not be able to retrieve the keys.If the keys are found then the cryptosystem is compromised.

For the substitution ciphers we have looked at, the size of the alphabet M is 26. Every symbol in the ciphertext C will be deciphered to become one of 26 possible symbols.Statistical

analysis is easy, we can use letter frequency and letter pattern frequency to find the key (or enough of the key to be able to read the message). Most cryptosystems in use these days are **permutation ciphers**.Text is first encoded using ASCII and then written in binary notation.The binary message is written in blocks of *b* bits.There are 2^b possible blocks and this is the size of the alphabet.The block is encrypted to another *b* bit block, so the ciphertext alphabet also has size 2^b.[15]

ASCII (American Standard Code for Information Interchange)

A	65	a	97	space	32	0	48
B	66	b	98	!	33	1	49
C	67	c	99	%	37	2	50
.		.		(	40	.	
.		.		)	41	.	
Z	90	z	122	,	44	9	57

In Caesar's cipher there are 26 possible keys. So the size of the keyspace is 26.For the substitution cipher there are *26!* ("26 factorial" = 26*25*24*…*2*1) possible keys which is approx. equal to 4×10^{26} but statistical analysis can make short work of this.A key length of 56 bits used to be secure (20 years ago) so the size of the key space was 2^{56}.These days a search through 2^{56} keys is computationally feasible.Keys are now of lengths 128, 192 or 256 bits. Suppose the key is k bits long. Then the key space has size 2^k.On average, Charles will have to investigate half of the keys until he finds the correct one = $2^k \div 2 = 2^{k-1}$.Suppose he can investigate N keys in a microsecond (N might be between 1 and a million depending on the information he has and the speed and number of computers).

Then Charles will take $2^{k-2} \div N$ microseconds to find the key.

	N=1	$N=10^6$
K=32	36 minutes	2 milliseconds
K=56	1142 years	10 hours
K=128	$5.4*10^{24}$ years	$5.4*10^{18}$ years

As we have already noted, the security of a cryptosystem is embodied in the values of the encryption and decryption keys. A cryptosystem is called ***symmetric*** if either key can be determined "easily" from knowledge of the other.Caesar's cipher and the substitution cipher are examples of symmetric cryptosystems.

Key Management Issues are [15]:

A. Key Generation:

Where are the keys generated and by whom? Perhaps Alice generates the keys and sends one to Bob (or vice versa) or may be a *Trusted Third Party* (TTP) generates the keys for them.

How are the keys generated? Is there a secure method to generate a key between Alice and Bob, or are the keys just a random stream?

B. Key Storage:

Where are the keys held once they have been generated?

C. Key Distribution:

How are the keys distributed to Alice and Bob (from each other or from the TTP). The channel they are using to communicate is insecure so they cannot send the keys over this channel.

D. Key Replacement:

How often are the keys replaced? In some applications, a key is used only once. In other circumstances, the key may be used for a time period of one second or perhaps one day.A key with a limited life is called a ***session key.***

Chaining Key are keys when Alice generates a new session key, and sends it to Bob first encrypting it with the old session key.The major problem with this technique is, if Charles discovers one key then he will be able to determine all subsequent keys.Random numbers are very important in cryptography. For example keys are often strings of random binary bits.

The random numbers are generated as :

a. Ideally by flipping a fair coin, but in reality by a computer programme.
b. Such numbers are only *pseudo-random*.

A random number generator uses some function *f* to generate a list of random numbers within a given range.Typically the next random number depends in some way on the previous one so that $r_{n+1} = f(r_n)$.The function *f* must be kept secret.

Suppose there are 3 people communicating using a symmetric key system, Alice, Bob and Dave. Each pair of people will need a separate pair of keys. So there will be 3 pairs of keys. If a fourth person, Emma, joins the group, then she will need to have a pair of keys for each of the other 3 people. So now we have 6 pairs of keys.

If there are *n* people communicating using a symmetric cryptosystem, and each pair of people share a key pair, then there will be a total of ***n*(n-1) / 2** pairs of keys required.

So for 10 people - 45 key pairs

For 100 people - 4,950 key pairs

For 1000 people - 499,500 key pairs

A random stream of binary bits is generated which is longer than the plaintext (also in binary bits). Alice and Bob each have the random stream - this is the key. The message is encrypted by XORing the plaintext with the key and decrypted in the same way. The key is only used once.This is method is known as one time padding.The One-Time Pad offers perfect secrecy since an interceptor can only guess whether or not any bit in the ciphertext was changed or not. Each bit is encrypted independently of all the other bits. The key cannot be guessed and knowledge of any part of the key does not help a cryptanalyst to discover any other part of the key.

How do Alice and Bob manage to each have the same random keystream?

The one-time pad is a kind of stream cipher - the plaintext is enciphered bit by bit by adding the **keystream** to the plaintext. The problem is that since the keystream for the one-time pad is random, it cannot be generated simultaneously by

both the sender and receiver. A more practical stream cipher uses a short key to generate a long keystream[15].

Example

a. Start with any binary key of length *n* and generate the next bit of the key stream by XORing the first and last bit of the previous *n* bits.
b. Depending on the key you start off with, it is possible to generate a stream which does not repeat until it has produced a keystream of length $2^n - 1$ bits.

For the i^{th} bit in any message:

$C_i = P_i \oplus K_i$

which means that:

$P_i = C_i \oplus K_i$

and

$K_i = P_i \oplus C_i$

If Charles knows a section of plaintext and ciphertext then he can easily find the key for that section.Thus security for a stream cipher relies on the design of the key stream generator.A keystream must be unpredictable.Designing a good keystream generator is difficult and advanced mathematics is required.However, there are many applications for stream ciphers because of their speed of use, ease of implementation and the fact that one bit of corrupt ciphertext does not impact on the rest of the message[15].

For a block cipher, the plaintext bit-string is divided into blocks of a given size and the encryption algorithm acts on that block to produce a cryptogram block (usually) of the same size. Block ciphers can be used to provide confidentiality, data integrity, user authentication or as the keystream generator for a stream cipher.

A well designed block cipher should satisfy amongst other things[15]:

a. ***The diffusion property*** - a small change in the plaintext should produce an unpredictable change in the ciphertext. This will prevent a differential analysis attack

b. ***The confusion property*** - a key that is "nearly correct" should give no indication of this fact. This will make exhaustive key searching much harder.

c. Every bit of the ciphertext should depend on every bit of the key. This is the property of ***completeness.*** This prevents a "divide and conquer" attack where a cryptanalyst tries to determine part of the key independently of other parts.

III. NOTATIONS

Alice/Bob	Two clients who want to communicate with each other
Pr_a, Pr_b	Private keys of Alice and Bob respectively
Pu_a,Pu_b	Public keys of Alice and Bob respectively
B_{FS}	Beginnig number of a Fibonacci Series
N	Number of bits
E	Assimmetric Encryption Scheme
D	Assimmetric Decryption Scheme
X, Y	Ciphertext of Pair (B_{FS} , N)

Figure 1: Notaions used in Proposed Cryptosystem

IV. PROPOSED CRYPTOSYSTM

A novel RSA based key atuhenticated two-party Prime Fibonacci Cryptosystem has been described as follows:

Note: B_{FS} starts from 3.

Step 1: Alice A Selects (B_{FS}, N) and computes Ciphertext of it by using Bob's public key as (X, Y) = E_{Pua} (B_{FS}, N).

Step 2: Now, Alice A Encrypts (X, Y) by using her private key Pr_a. i.e., (U, V) = Pr_a (X, Y) and sends this message to B.

Step 3: Now Bob B, after receiving this message decrypts it by using Alice A's public key (X,Y)=D_{Pua}(U,V) for authenticating that received message came from A.

Step 4: Now Bob B generates the original pair (B_{FS}, N) by decrypting the (X, Y) by using his own private key Pr_b. i.e., (B_{FS}, N)= Pr_b(X,Y).

Step 5: Now Bob B generates fibonacci series of N bits starting from $B_{FS.}$ i.e., ($B_{FS,}$ $N_{FS1,}$ N_{FS2}..... N_{FSn} (N times)).

Step 6: Follow the Markel-Hellman easy knapsack procedure.

A. Markel-Hellman easy knapsack:

The Markle-Hellman knapsack cipher encrypts a message as a knapsack problem. The plaintext block transforms into binary string (the length of block is equal number of elements in knapsack sequence)[12].One value determines that an element will be in target sum. This sum is a ciphered message. Table I shows an example of solving the knapsack problem for the entry numbers sequence:

Table 1: Example of Knapsack Encryption

Plaintext	Knapsack Sequence	Ciphertext
1 0 1 0 0	3 5 13 89 233	3+13=26
0 1 1 0 1	3 5 13 89 233	5+13+233=251
1 0 0 1 1	3 5 13 89 233	3+89+233=325

Easy knapsacks have a sequence of numbers that are superincreasing - that is, each number is greater then the sum of previous numbers: $a_i > \sum_{j=1}^{i-1} a_i$ for i=2,……, n(where *i a* is i-the element of the sequence)[12]. For example {3,5,13,89,233} is a superincreasing sequence but {3,5,13,89,233} is not. The knapsack solution with the superincreasing sequence proceeds as follows. The target sum is compared with a greatest number in the sequence. If the target sum is smaller, than this number, the knapsack will not fill, otherwise it will. Then the smaller element is subtracted from the target sum, and the result of the subtraction, is compared with next element. Such operation is done until the smallest number of sequence is reached. If the target sum is reduced to 0 value, than solution exists. In other case solution doesn't exist. For example, consider a total knapsack target sum is 251 and the sequence of weights of {3, 5, 13, 89, 233}.

The largest weight, 233, is less then 251, so 233 are in the knapsack, Subtracting 233 from 251 leaves 18. The next number 89 is greater than 18, so 89 is not in the knapsack. The next weight 13 is less than 18, hence, 13 is in the knapsack, so subtracting 13 from 18 leaves 5. The next weight 5, is equal to 5, so 5 are in the knapsack and the total weight is brought to 0, which indicates that a solution has been found. The plaintext

that resulted from a ciphertext value of 233 would be 01101. The superincreasing knapsack is easy to decode, which means that it does not protect the data. Anyone can recover the bit pattern from the target sum for a superincreasing knapsack if the elements of the superincreasing knapsack are known.

The knapsack solution with the super increasing sequence proceeds as follows. The encrypted text (cipher) is known as target sum. Decryption is done as follows. The target sum is compared with a largest number in the sequence. If the target sum is smaller, than this number, the knapsack will not fill, otherwise it will. Then the smaller element is subtracted from the target sum, and the result of the subtraction, is compared with next element. The corresponding bit of the knapsack sequence is 1 if the knapsack element is subtracted from the target sum. Otherwise the bit is 0. Such operation is done until the smallest number of sequence is reached. If the target sum is reduced to 0 value, than solution exists. In other case solution doesn't exist.

Table 2: Example of Knapsack Encryption

Ciphertext	Knapsack Sequence	Plaintext
26	3 5 13 89 233	1 0 1 0 0
251	3 5 13 89 233	0 1 1 0 1
325	3 5 13 89 233	1 0 0 1 1

V. RESULTS AND DISCUSSION

Alice (Pr_a, Pu_a) Bob (Pr_a, Pu_a)

1. $(X, Y) = E_{pub}(B_{FS}, N)$
2. $(P, Q)=E_{pra}(X,Y)$

3. $D_{pua}(E_{pra}(X,Y))=(X,Y)$
4. $D_{prb}(X,Y)=(B_{FS}, N)$

Figure 2: Steps in Proposed Cryptosystem

Let us consider an example, where Alice A selects a start verctor B_{FS}=3 and the number of binary digits N= 5 in a stream cipher.

Using the above example Alice A Encrypts (3,5) with public key of Bob B.And then again Alice A encrypts the result with her private key(pr_a) and sends the result to Bob B.After receiving the vector, Bob B Decrypts the encrypted vector with the public key of Alice A.Then again decrypts it with his private key(pr_b) and retrieves vector (3,5).

Now Bob B generates the knapsack as follows: <3, 5, 13, 89, 233>.

Let's try to send a message that is in binary code: 1 0 1 0 00 1 1 0 11 0 0 1 1.

Prime Fibonacci sequence(PFS) and code		Output	Mark 1 in the corresponding output elements and remaining with 0 in a PFS vector <3, 5, 13, 89, 233>
<3, 5, 13, 89, 233>	26	3,13	10100
<3, 5, 13, 89, 233>	251	5,13,233	01101
<3, 5, 13, 89, 233>	325	3,89,233	10011

Figure 3: Example of Proposed Cryptosystem

The knapsack contains five weights so we need to split the message into groups of five each:

1 0 1 0 0
0 1 1 0 1
1 0 0 1 1

This corresponds to three sets of weights with totals as follows:

10100 = 3 + 13 = 26
01101=5+13+233=251
10011 = 3+89+233 =325
So the coded message is 26 251 325.

Now Alice A will decrypt the received message as follows:
Alice A has receive the message as 26 251 325.

Hence by arranging we get the original message. The decoded message is: 101000110110011.

VI. CONCLUSION

This paper presents a new and efficient RSA based key Authenticated Prime Fibonacci Cryptosystem.In this scheme two clients can communicate each other securely by generating the prime fibonacci series (super-increasing order) by with the exchange of only the pair consisting of start and the limit of the fibonacci series.Unlike others existing systems this crypto system is completely new scheme based on fibonacci series.

Simple and short fibonacci series like the one presented here is of no use in real implementations. They are too easy to break. Practical implementations should contain at least 500 terms (items). Each term in the super-increasing order should be 500 bits long. Hence fibonacci series of this size are infeasible to solve by brute force.

VII. ACKNOWLEDGMENT

The research reported in this paper is the result of a team effort that has started shortly. Dozens of researchers from all over the world published many papers on this subject, and we have greatly benefited from their ingenious ideas and beautiful insights. They are too numerous to list here, but we are gratefully acknowledge the contribution of all of them.

VIII. REFERENCES

[1]. W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. IT-22, pp. 644{654, Nov. 1976.

[2]. S. M. Bellovin and M. Merrit, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in: IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, 1992, pp. 72-84.

[3]. D. Seo and P. Sweeney, "Simple authenticated key agreement algorithm," IEE Electronics Letters, vol. 35, no. 13, pp. 1073-1074, 1999.

[4]. Y. M. Tseng, "Weakness in simple authenticated key agreement protocol," IEE Electronics Letters, vol. 36, no. 1, pp. 48-49, 2000.

[5]. W. C. Ku and S. D. Wang, "Cryptanalysis of modified authenticated key agreement protocol," IEE Electronics Letters, vol. 36, no. 21, pp. 1770-1771,2000.

[6]. Steiner, G. Tsudik and M. Waidner, "Refinement and extension of encrypted key exchange," ACM Operating Systems Review, 29 (1995) 22-30.

[7]. C. L. Hsu, T. S. Wu, T. C. Wu, and C. Mitchell, "Cryptanalysis of enhancement for simple authenticated key agreement algorithm," Applied Mathematics and Computation, vol. 142, no. 2-3, pp. 305-308,2003.

[8]. C. C. Chang, K. F. Hwang, and I. C. Lin, "Security enhancement for a modified authenticated key agreement protocol," International Journal of Computational and Numerical Analysis and Applications, vol. 3, no. 1, pp. 1-7, 2003.

[9]. C. C. Chang, K. F. Hwang, and I. C. Lin, "Security enhancement for a modified authenticated key agreement protocol," International Journal of Computational and Numerical Analysis and Applications, vol. 3, no. 1, pp. 1-7, 2003.

[10]. Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," ACM Operating Systems Review, 29 (1995) 77-86.

[11]. Jung-Wen Lo, Shu-Chen Lin and Min-Shiang Hwang "A Parallel Password-Authenticated Key Exchange Protocol For Wireless Environments**"** ISSN 1392 – 124X Information Technology And Control, 2010, Vol.39, No.2

[12]. Poonam Garg, Aditya Shastri, and D.C. Agarwal "An Enhance Cryptanalytic Attack on knapsack cihper using Genetic Algorithms" ISSN 1307-6884, 12 MARCH 2006 PWASET VOLUME .

[13]. Ya-Fen Chang, Wei-Cheng Shiao, and Chung-Yi Lin "Comments on Yoon and Yoo's Three-party Encrypted Key Exchange Protocol" 2009 International Conference on Advanced Information Technologies (AIT).

[14]. R.Padmavathy "IMPROVED THREE PARTY EKE PROTOCOL" ISSN 1392 – 124X Information Technology And Control, 2010, Vol.39, No.3.

[15]. Dr.Rachel Shipsey, Computer Security CIS326,PPT, "http://homepages.gold.ac.uk/rachel/#CIS326".