International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Secure Digital Signature Scheme against Chosen Message Attacks

Maheshkumar Zilla [1], K. Raghavendra Rao [2]

[1]M.Tech student, Department of CSE, Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE, Anurag Group of Institutions, Hyderabad, India

Abstract: ***Digital signature scheme based on the computational difficulty of integer factorization. The scheme possesses secure against an adaptive chosen-message attack: Receiver who receives signatures for messages of his choice (where each message may be chosen in a way that depends on the signatures of previously chosen messages) cannot later forge the signature. This is surprising, since the properties of having forgery being equivalent to factoring and being invulnerable to an adaptive chosen-message attack were considered in the folklore to be contradictory. More commonly, we show how to construct a signature scheme with such properties based on the existence of a "claw-free" pair of permutations – a potentially weaker assumption than the intractability of integer factorization. The new scheme is potentially realistic: signing and verifying signatures are practically fast, and signatures are compact.***

Keywords: Cryptography, digital signatures, factoring, chosen-message attacks, authentication, trap-door permutations, randomization.

1. Introduction

The idea of a "digital signature" first appeared in Diffie and Hellman's seminar paper that is "New Directions in Cryptography". According to that each user can publish a public key, while keeping secret a secret key. In their scheme user A generates a signature for the message M which is depends on M and on A's secret key, such that anyone can verify the validity of signature using A's public key. However, while knowing user A's public key is enough to allow one to validate A's signatures, it does not allow anyone to easily forge A's signatures. They also proposed the way of implementing signatures based on "trap-door functions".

The notion of a digital signature is useful and is a legal replacement for handwritten signatures [7, 8]. However, a number of technical problems arise if digital signatures are implemented using trap-door functions as suggested by Diffie and Hellman [2]; these problems have been addressed and solved in part elsewhere. Consider, [5] showed how to handle s sparse messages sets and how to ensure that if an enemy sees previous signatures (for messages that he has not chosen) it does not help him to forge new signatures (this is a "non-adaptive chosen-message attack").

The signature scheme presented here, using totally different ideas than those presented by Diffie and Hellman, they advance the state of the art of signature schemes with security properties even further; it has the following important characteristics:

- What we prove that forgery is difficult and not merely obtaining the secret key used by the signing algorithm (or obtaining an efficient equivalent algorithm).
- Forgery is proven to be difficult for a "most general" enemy who can mount an adaptive chosen-message attack. In contrast to all previous published work on this problem, we prove that this scheme is invulnerable against such an adaptive attack where each message whose signature is requested may depend on all signatures previously obtained from the real signer. We believe that an adaptive chosen-message attack is the most powerful attack possible for an enemy who is restricted to using the signature scheme in a natural manner.
- The properties we prove about the new signature scheme do not depend in any way on the set of messages which can be signed or on any assumptions about a probability distribution on the message set.
- This scheme can be generalized so that it can be based on "hard" problems other than factoring whenever one can create claw-free trap-door pair generators.

Our scheme can be based on family of pairs of claw-free permutations, yielding a signature scheme that is invulnerable to a chosen-message attack even if the claw-free trap-door permutations are vulnerable to a chosen-message attack when used to make a trap-door signature scheme. Basic ideas in the construction are the use of randomization, signing by using two authentication steps (the first step authenticates a random value which is used in the second step to authenticate the message), and the use of a tree-like branching authentication structure to produce short signatures. We note that our signature scheme is not of the simple Diffie-Hellman "trap-door" type. Such as given message can have many signatures.

- The general technique for forging signatures can be used as a "black box" in a construction that enables the enemy to aspect one of the signer's public moduli.
- The technique of "forging" signatures by getting the real signer to play the role of the "black box" (i.e. getting the actual signer to produce some desired genuine signatures) does not help the enemy to factor either of the signer's moduli.

2. Fundamental Notions

To properly characterize the results of this paper, need to answer the following questions:

- What is a digital signature scheme?
- What kinds of attacks can the enemy made against a digital signature scheme?
- What is meant by "breaking" the signature scheme?

2.1 What Is a Digital Signature Scheme?

A digital signature scheme consists of following components:

- A security parameter k, which is chosen by the user when he creates his public and secret keys. Here the parameter k determines a number of quantities (length of signatures, length of signable messages, running time of the signing algorithm).
- A message space **M** which is the set of messages to which the signature algorithm may be applied.
- A signature bound B which is an integer bounding the total number of signatures that can be produced with an instance of the signature scheme.
- A key generation algorithm G which any user A can use on input 1^k (i.e. k in unary) to generate in polynomial time a pair (P_A^k, S_A^k) of matching public and secret keys.
- A signature algorithm σ which produces a signature $\sigma(M, S_A)$ for the given message M using the secret key S_A.
- A verification algorithm V which tests whether S is a valid signature for message M using the public key P_A. (I.e. V (S, M, P_A) will be **true** if and only if it is valid.)

Any of the above algorithms may be "randomized" algorithms that make use of auxiliary random bit stream inputs. Here G must be a randomized algorithm, since part of its output is the secret key, which is unpredictable to an adversary. The signing algorithm σ is randomized – our signing algorithm is randomized and is capable of producing many different signatures for the same message. Generally, the verification algorithm need not be randomized, and ours is randomized.

2.2 Kinds of Attacks

We differentiate two basic kinds of attacks:

- **Key-Only Attacks** here enemy knows only the signer's public key.
- **Message Attacks** where the enemy is able to examine some signatures corresponding to either known or chosen-messages before his attempt to break the scheme.

We further identify the following four kinds of message attacks, which are characterized by how the messages whose signatures the enemy chooses the messages and signatures.

- **Known Message Attack:** The enemy is given access to signatures for a set of messages $m_1, \ldots, m_t$. The messages are known to the enemy but are not chosen by him.
- **Generic Chosen Message Attack:** Here the enemy is allowed to obtain from A valid signatures for a chosen list of messages $m_1, \ldots, m_t$ before he breaks the A's signature scheme.
- **Directed Chosen Message Attack:** This is same as generic chosen-message attack, except that the list of messages to be signed may be created after seeing A's public key but before any signatures are seen.
- **Adaptive Chosen Message Attack:** This is more general one. Here the enemy is also allowed to use A as an "oracle"; not only may he request from A signatures of messages which depend on A's public key but he may also request signatures of messages which depend additionally on previously obtained signatures.

2.3 What Does It Mean To "Break" a Signature Scheme?

One might say that the enemy has "broken" user A's signature scheme if his attack allows him to do any of the following with a non-negligible probability:

- **A Total Break:** Compute A's secret trap-door information.
- **Universal Forgery:** Find an efficient signing algorithm functionally equivalent to A's signing algorithm (based on possibly different but equivalent trap-door information).
- **Selective Forgery:** Forge a signature for a particular message chosen a priori by the enemy.
- **Existential Forgery:** Forge a signature for at least one message. The enemy has no control over the message whose signature he obtains, so it may be random. Consequently this forgery may only be a minor nuisance to A.

Note that to forge a signature means to produce a new signature; it is not forgery to obtain from A a valid signature for a message and then claim that he has now "forged" that signature, any more than passing around an authentic handwritten signature is an instance of forgery. For example, in a chosen-message attack it does not constitute selective forgery to obtain from the real signer a signature for the target message M.

3. Previous Signature Schemes and their Security

Here we list a number of previously proposed signature schemes and briefly review some facts about their security.

Trap-Door Signature Schemes [2]: Any trap-door signature scheme is existentially forgeable with a key-only attack since a valid (message, signature) pair can be created by beginning with a random "signature" and applying the public verification algorithm to obtain the corresponding "message". A common heuristic for handling this problem in practice is to require that the message space be sparse (i.e. requiring that very few strings actually represent messages – for example this can be enforced by having each message contain a reasonably long checksum.) In this scheme this specific attack is not likely to result in a successful existential forgery.

Rivest-Shamir-Adleman [11]: The RSA scheme is selectively forgeable using a directed chosen-message attack, because RSA is multiplicative: the signature of a product is the product of the signatures. (It can be handled in practice as above using a sparse message space.)

Merkle-Hellman [7]: Shamir showed the basic Merkle-Hellman "knapsack" scheme to be universally forgeable using just a key-only attack [13]. (This scheme was perhaps more an encryption scheme than a signature scheme, but it had been proposed for use as a signature scheme as well.)

Rabin [10]: Rabin's signature scheme is totally breakable if the enemy uses a directed chosen-message attack. However, for non-sparse message spaces selective forgery is as hard as factoring if the enemy is restricted to a known message attack.

Williams [15]: This scheme is same as Rabin's. The proof that selective forgery is as hard as factoring is a little stronger, since here only a single instance of selective forgery guarantees factoring (Rabin needed a probabilistic argument). Williams uses effectively (as we do) the properties of numbers which are the product of a prime $p \equiv 3 \pmod 8$ and a prime $q \equiv 7 \pmod 8$. Again, this scheme is totally breakable with a directed chosen-message attack.

Lieberherr [6]: This scheme is similar to Rabin's and Williams', and is totally breakable with a directed chosen-message attack.

Shamir [12]: This knapsack-type signature scheme has recently been shown by Tulpan [14] to be universally forgeable with a key-only attack for any practical values of the security parameter.

Goldwasser-Micali-Yao [5]: This paper presents for the first time signature schemes which are not of the trap-door type, and it have the interesting property that their security characteristics hold for any message space. The first signature scheme in [5] was proven not to be even existentially forgeable against a generic chosen-message attack unless factoring is easy. However, it is not known to what extent directed chosen-message attacks or adaptive chosen-message attacks might aid an enemy in "breaking" the scheme**.** The second scheme presented there (based on the RSA function) was also proven not to be even existentially forgeable against a generic chosen-message attack. This scheme may resist existentially forgery against an adaptive chosen-message attack, even though this has not been proven. (A proof would require showing certain properties about the density of prime numbers and making a stronger intractability assumption about inverting RSA.) We may note that, by comparison, scheme presented here is so much faster, produces much more compact signatures, and it is based on much simpler assumptions (only the difficult of factoring or more generally the existence of claw-free permutation pair generators).

Several ideas and techniques presented in [5], such as bit-by-bit authentication, are used in the present paper.

Ong-Schnorr-Shamir [9]: Totally breaking this scheme using an adaptive chosen-message at-tack has been shown to be as hard as factoring. However, Pollard has recently been able to show that the "OSS" signature scheme is universally forgeable in practice using just a key-only attack; he developed an algorithm to forge a signature for any given message without obtaining the secret trap-door information. A more recent "cubic" version has recently been shown to be universally forgeable in practice using just a key-only attack (also by Pollard). An even more recent version based on polynomial an equation was similarly broken by Estes, Adleman, Kompella, McCurley and Miller for quadratic number fields.

El Gamal[4]: This scheme, based on the complexity of computing discrete logarithms, is existentially forgeable with a generic message attack and selectively forgeable using a directed chosen-message attack.

Okamoto-Shiraishi[8]: This scheme is based on the difficulty of solving quadratic inequalities mod-ulo a composite modulus, was shown to be universally forgeable by Brickell and DeLaurentis [1].

4. The Paradox of Proving Signature Schemes Secure

The paradoxical nature of signature schemes which are provably secure against chosen-message at-tacks made its first appearance in Rabin's paper, "Digitalized Signatures as Intractable as Factorization" [10]. The signature scheme proposed there works as follows. User A publishes a number n which is the product of two large primes. To sign a message M , A computes as M 's signature one of M 's square roots modulo n. (When M is not a square modulo n, A modifies a few bits of M to find a "nearby" square.) Here signing is essentially just extracting square roots modulo n. Using the fact that extracting square roots modulo n enables one to factor n, it follows that selective forgery in Rabin's scheme is equivalent to factoring if the enemy is restricted to at most a known message attack.

However, it is true (and was noticed by Rabin) that an enemy might totally break the scheme using a directed chosen-message attack. By asking A to sign a value $x^2 \bmod n$ where x was picked at random, the enemy would obtain with probability $\frac{1}{2}$ another square root y of x^2 such that $\gcd(x + y, n)$ was a prime factor of n.

Rabin suggested that one could overcome this problem by, for example, having the signer concatenate a fairly long randomly chosen pad U to the message before signing it. In this way the enemy can not force A to extract a square root of any particular number.

However, the reader may now observe that the proof of the equivalence of selective forgery to factoring no longer works for the modified scheme. That is, being able to selectively forge no longer enables the enemy to directly extract square roots and thus to factor. Of course, breaking this equivalence was really the whole point of making the modification.

4.1 The Paradox

We now "prove" that it is impossible to have a signature scheme for which it is both true that forgery is provably equivalent to factoring, and yet the scheme is invulnerable to adaptive chosen-message attacks. The argument is essentially the same as the one given in [15]. By forgery we mean in this section any of universal, selective, or existential forgery – we assume that we are given a proof that forgery of the specified type is equivalent to factoring.

Let us begin by considering this given proof. The main part of the proof presumably goes as follows: given a subroutine for forging signatures, a constructive method is specified for factoring. (The other part of the equivalence, showing that factoring enables forgery, is usually easy, since factoring

usually enables the enemy to totally break the scheme.)

But it is trivial then to show that an adaptive chosen-message attack enables an enemy to totally break the scheme. The enemy merely executes the constructive method for factoring given in the proof, using the real signer instead of the forgery subroutine! That is, whenever he needs to execute the forgery subroutine to obtain the signature of a message, he merely performs an "adaptive chosen-message attack" step – getting the real user to sign the desired message. In the end the unwary user has enabled the enemy to factor his modulus! (If the proof reduces factoring to universal or selective forgery, the enemy has to get the real user to sign a particular message. If the proof reduces factoring to existential forgery, the enemy need only get him to sign anything at all.)

4.2 Breaking the Paradox

How can one hope to get around the apparent contradictory natures of equivalence to factoring and invulnerability to an adaptive chosen-message attack?

The key idea in resolving the paradox is to have the constructive proof that forgery is as hard as factoring be a uniform proof which makes essential use of the fact that the forger can forge for arbitrary public keys with a non-negligible probability of success. However, in "real life" a signer will only produce signatures for a particular public key. Thus the constructive proof cannot be applied in "real life" (by asking the real signer to unwittingly play the role of the forger) to factor.

In our scheme this concept is implemented using the notion of "random rooting". Each user publishes not only his two composite moduli n_1 and n_2, but also a "random root" r. This value r is used when validating the user's signatures. The paradox is resolved in our case as follows:

It is provably equivalent to factoring for an enemy to have a uniform algorithm for forging; uniform in the sense that if for all pairs of composite numbers n_1 and n_2 if the enemy can randomly forge signatures for a significant fraction of the possible random roots r, then he can factor either n_1 or n_2.

The above proof requires that the enemy be able to pick r himself – the forgery subroutine is fed triples (n_1, n_2, r) where the r part is chosen by the enemy according the procedure specified in the constructive proof. However, in "real life" the user has picked a fixed r at random to put in his public key, so an adaptive chosen-message attack will not enable the enemy to "forge" signatures corresponding to any other values of r. Thus the constructive method given in the proof cannot be applied! More details can be found in section 9.

5. Building Blocks for Signing

In this section we define the basic building blocks needed for describing our signature scheme. In later section, we will define what a signature is and how to sign, using the objects and data structures specified here.

Assumption: Assume from here on that all claw-free functions used are defined over domains which do not include the empty string. This assumption is necessary because we use as a "marker" in our construction; note that it is easy, via simple recordings', to enforce this construction if necessary. We start by defining the essential notion of an f -item.

Definition: Let $f = (d_f, f_0, f_1)$ be a claw-free pair. A tuple of strings $(t, r; c_1, \ldots, c_m)$ is an f -item if

$$f_{hc_1,\ldots,c_m\, i}(t) = r$$

Definition: In an f -item $(t, r; c_1, \ldots, c_m)$,

- t is called the tag ,
- r is called root of the item
- The c_i 's are the children belong to item. We note that the children are ordered, so that we can speak of the first child or the second child of the item.

Reminder that given a claw free pair f and a tuple it is easy to check if the tuple is an f -item by applying the appropriate f_{hii} to the tag, and checking if the correct root is obtained. Figure 1 shows our graphic representation of an f -item $(t, r; c_1, c_2)$ with two children.

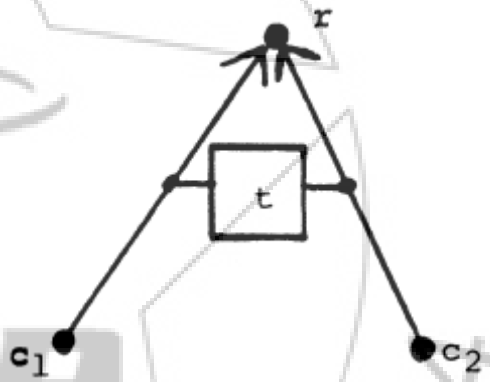

Figure 1: An f -item with two children

Definition: We say that a sequence of f -items $L_1, L_2 \ldots L_b$ is an f -chain starting at y if, for $i = 1, \ldots, b - 1$, the root of L_{i+1} is one of the children of L_i and y is the root of L_1. We declare the chain ends at x if x is one of the children of the item L_b. For efficiency related considerations, our signature scheme will organize a collection of a special type of f -chains in the tree-like structure defined below.

Definition: Let i be a binary string of length b and f a claw-free pair. An f -i-tree is a bijection T between DF S(i) and a set of f -items such that:

(1) if string j has length b, then T (j) is an f -item with exactly two children, exactly one of which is , the empty string. These f -items are called bridge items.

(2) if string j has length less than b, then T (j) is an f -item with exactly two children, c_0 and c_1, both of which are non-empty strings. Moreover, c_0 , the 0th child, is the root of T (j0) and c_1, the 1st child, the root of T (j1).

The f -item T (j) is said to be of depth d if string j has length d. (The bridge items are thus the items of b depth.) The root of T is the root of the f -item T ( ). The internal nodes of T are the root and the children of the f -items of depth less than

b. The leaves of the T are the non-empty children of the bridge items. Thus the internal nodes and the leaves of an f -i-tree are actual values and not f -items. Leaves possess binary names of length b, leaf j is non-empty child of bridge item T (j). The path to leaf j = $j_0 \ldots j_b$ is the f -chain T ( ), T (j_0), . . . , T ($j_0 \ldots j_b$).

Figure 2 gives our graphic representation of an f -100-tree, as it would be used in our signature scheme. In this figure we denote by r_i^f the root of f -item T (i), and by r_i^g the leaf (non-empty) child of bridge item T (i). (Also present in this figure are a number of "g-items", which are not part of the f -100 tree but are attached to it in a manner to be described.

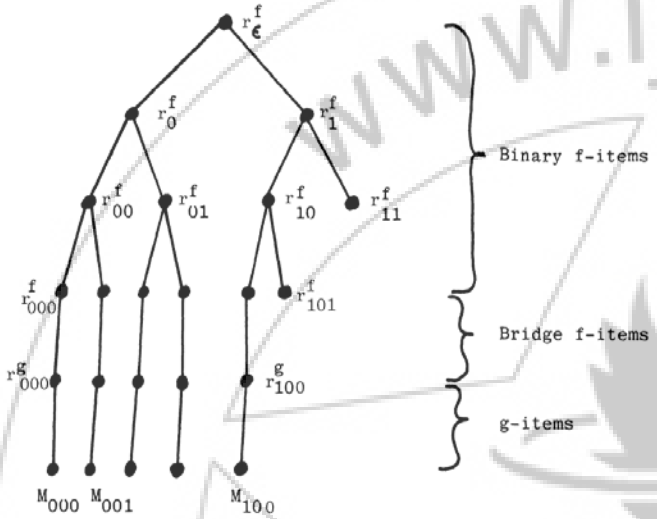

Figure 2: An f- lO0-tree.

There are two reasons for letting the bridge items of an f -i-tree have the empty string as one of their children. First, it makes them de facto f -items with single child, a subtle point in our proof of security that is pointed in remark 1. Second, it makes them distinguishable from items with two children that is the simple point used.

6. Description of our Signature Scheme

6.1 Message Spaces

The security properties of the signatures scheme hold for any nonempty message space M $\subset \{0, 1\}^+$.

6.2 How to Generate Keys

Assume the existence of a claw-free permutation pair generator G and, without loss of generality, that the bound B on the number of signatures that can be produced is a power of 2:
B = 2^b.

The key-generation algorithm K running shown below on inputs 1^k and 2^b:
1) K runs G twice on the input 1^k to secretly and randomly select two quintuples
2) (d_f , f_0, f_0^{-1}, f_1, f_1^{-1}), and (d_g , g_0, g_0^{-1}, g_1, g_1^{-1}) $\in$ [G(1^k)].
3) K then randomly selects r^f in D_f = [d_f ()].
4) K outputs the public key P K = (f, r^f , g, 2^b) where f is the claw-free pair (d_f , f_0, f_1) and g is the claw-free pair (d_g , g_0, g_1).
5) K outputs the secret key SK = (f^{-1}, g^{-1}).
6) The P K and SK so produced are said to be (matching) keys of size k.

6.3 What Is a Signature?

A signature of a message m with respect to a public key (f, r^f , g, 2^b) consists of:
1) An f -chain of length b + 1 starting at a string r^f and ending at r^g , and
2) A g-item with r^g as its root and m as its only child.

6.4 How to Sign?

In the remainder of this section we shall presuppose that user A's public key is P K = (f, r^f , g, 2^b) where f = (d_f , f_0, f_1) and g = (d_g , g_0, g_1). User A's secret key is SK = (f^{-1}, g^{-1}). We denote by D_f the domain [d_f ()], and denote by D_g the domain [d_g ()] similarly. Conceptually, user A creates an f -1^b-tree T , which has 2^b leaves. The root of T will be r^f . The other internal nodes of T are randomly selected elements of D_f . The leaves of T are randomly selected elements of D_g .

To sign m_i, the i − th message in the chronological order, user A computes a g-item G_i whose root $r_i^g \in D_g$ is the ith leaf of T , and whose only child is the message m_i . He then outputs, as the signature of m_i, G_i and the f -chain in T starting at root r^f and ending at leaf r_i^g .

In practice, it will be undesirable for user A to precompute and store all of the T. He will instead" grow" T as needed and try to optimize his use of storage space and time. This is taken into account by our signing procedure. We describe a variation of our signing method that requires the signer to remember just his secret key and his most recently produced signature, in order to produce his next signature. The reader may find it helpful to refer to Figure 2 while reading this description.

6.5 How to Verify a Signature

Given A's public key (f, r^f , g, 2^b), anyone can easily verify that the first b+1 elements in the signature of m_i are f -items forming an f -chain starting at r^f and ending at r_i^g , and that the g-item in the signature has r_i^g as its root and m_i as its only child. If these checks are all satisfied, the given sequence of items is accepted as an authentic signature by A of the message m_i. It is easy to confirm that these operations take time proportional to b times some polynomial in k, the size of the public key.

7. Efficiency of the Proposed Signature Scheme

Assume that if f = (d_f , f_0, f_1) is a claw-free pair of size k, then an element of D_f is specified by a k-bit string. Then the time to compute a signature for a message m of length l is is O(bk) f -inversions (i.e. inversions of f_0 or f_1) and O(l) g-inversions. Another relevant measure of efficiency is "amortized" time. That is, the time used for producing all possible 2^b signatures

divided by 2 [b]. In our scheme, the amortized "f -inversion" cost is O(k). The amortized "g-inversion" cost is O(l) if the average length of a message is l. The length of the signature for m is O(bk + l), where l is the length of m, as m is included in m's signature as the child of the g-item. Clearly, if m is known to the signature recipient, the g-item need not include m: it suffices to give its root and its tag. This way the length of the signature can be only O(bk) long, which is independent of the length of m and possibly much shorter. The memory required by the signing algorithm is O(bk) since it consists of storing (the f -items in) the most recently produced signature.

8. Conclusion

This scheme possesses the novel property of being robust against an adaptive chosen-message attack: an adversary who receives signatures for messages of his choice (where each message may be chosen in a way that depends on the signatures of previously chosen messages) cannot later forge the signature of even a single additional message.

References

[1] Brickell, E., and J. DeLaurentis, "An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi," Proc. CRYPTO 85 (Springer 1986).
[2] Diffie, W. and M. E. Hellman, "New Directions in Cryptography", IEEE Trans. Info. Theory **IT-22** (Nov. 1976), 644-654.
[3] Estes, D., L. Adleman, K. Kompella, K. McCurley, and G. Miller, "Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number Fields," Proc. CRYPTO 85, to appear.
[4] El-Gamal, T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", Proceedings of Crypto 84 (Springer 1985), 10–18.
[5] Goldwasser, S., S. Micali, and A. Yao, "Strong Signature Schemes," Proc. 15th Annual ACM Symposium on Theory of Computing, (Boston Massachusetts, April1983), 431-439.
[6] Lieberherr, K. "Uniform Complexity and Digital Signatures," Theoretical Computer Science **16**,1 (Oct. 1981), 99-110.
[7] Merkle, R., and M. Hellman, "Hiding Information and Signatures in Trap-Door Knapsacks," IEEE Trans. Infor. Theory **IT-24** (Sept. 1978), 525-530.
[8] Okamoto, T., and A. Shiraishi, "A Fast Signature Scheme Based on Quadratic In- equalities," Proc. 1985 Symp. on Security and Privacy (Oakland, April 1985).
[9] Ong, H., C. Schnorr, and A. Shamir, "An Efficient Signature Scheme Based on Quadratic Equations," Proc. 16th Annual ACM Symposium on Theory of Computing, (Washington, D.C., April 1984), 208-217.
[10] Rabin, Michael. "Digitalized Signatures as Intractable as Factorization," MIT Lab- oratory for Computer Science Technical Report MIT/LCS/TR-212 (Jan. 1979).
[11] Rivest, R., A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. of the ACM (Feb. 1978), 120-126.
[12] Shamir, A., "A Fast Signature Scheme," MIT Laboratory for Computer Science Technical Memo MIT/LCS/TM-107 (July 1978).
[13] Shamir, A., "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," Proc. 23rd Annual IEEE FOCS Conference (Nov. 1982), 145-152.
[14] Tulpan, Y., "Fast Cryptanalysis of a Fast Signature System," Master's Thesis in Applied Mathematics, Weizmann Institute. (1984).
[15] Williams, H. C., "A Modification of the RSA Public-Key Cryptosystem," IEEE Trans. Info. Theory **IT-26** (Nov. 1980), 726-729.

Author Profile

Maheshkumar Zilla received the B.Tech degree in computer science and Engineering from JNTU Hyderabad in 2012 and pursuing M.tech. degree in Computer science and Engineering from JNTU Hyderabad, India

K. Raghavendra Rao working as assistant professor in Computer Science Engineering from CVSR College of Engineering from Anurag Group of Institutions Venkatapur(V), Ghatkesar(M), Ranga Reddy District, Hyderabad-500088, Telangana State, India