Improved ID Based Key Agreement Protocol Using Timestamp

M.D.P Kishore , Dr. K.Venkata Rao

Department of Computer Science &Engineering
Vignan's Institute of Information Technology
Visakhapatnam, India

Abstract: **ID-based encryption (or identity-based encryption (IBE)) is an important primitive of ID-based cryptography. As such it is a type of public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address). This can use the text-value of the name or domain name as a key or the physical IP address it translates to. Authenticated key agreement protocols are fundamental for secure communication over insecure environments. However design of a secure protocol is erroneous because of the inherent complexity of the problem. In this paper we present a novel improvement to the existing improved Lee-Lee Identity based key agreement protocol using the technique of Timestamp. We have shown that popular analysis frameworks (Lee Lee models) can be extended in a natural way using this technique, and that this permits addressing a new class of real-world protocols that, until now, lacked a complete formal treatment. In order to rigorously analyze the security of this protocol, one would need to define a timed version of three-party key agreement security models. Moving away from key agreement and authentication protocols, our approach opens the way for the formal analysis of time-related cryptographic protocols such as those aiming to provide secure message time-stamping and clock-synchronization. Finally, it would be interesting to see how one could apply a similar approach to security models that try to capture public key infrastructures.**

Keywords–Cryptography, Key Agreement protocol,Timestamp

1. INTRODUCTION

In cryptography, a key-agreement protocol is a protocol whereby two or more parties can agree on a key in such a way that both influence the outcome. If properly done, this precludes undesired third-parties from forcing a key choice on the agreeing parties. Protocols that are useful in practice also do not reveal to any eaves dropping party what key has been agreed upon.

Two party key agreement protocols allow communicating parties to establish a common secret key via public communication. The agreed key i.e. the session key can then be used to establish a secure communication channel between the two parties.

The first publicly known public-key agreement protocol that meets the above criteria was the Diffie-Hellman [1] exponential key exchange, in 1976 , in which two parties jointly exponentiate a generator with random numbers, in such a way that an eavesdropper has no way of guessing what the key is.

However, exponential key exchange in and of itself does not specify any prior agreement or subsequent authentication between the participants. It has thus been described as an anonymous key agreement protocol. Since then many key agreement protocols have been proposed with different securities properties.

ID-based encryption

ID-based encryption (or identity-based encryption (IBE)) is an important primitive of ID-based cryptography. As such it is a type of public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address). This can use the text-value of the name or domain name as a key or the physical IP address it translates to.

The first implementation of an email-address based PKI was developed by Adi Shamir in 1984,[1] which allowed users to verify digital signatures using only public information such as the user's identifier.

ID-based encryption was proposed by Adi Shamir in 1984.[1] He was however only able to give an instantiation of identity-based signatures. Identity-based encryption remained an open problem for many years. One example of the research leading up to identity-based encryption is provided in Maurer.[2]

The pairing-based Boneh–Franklin scheme[3] and Cocks' encryption scheme[4] based on quadratic residues both solved the IBE problem in 2001.

Timestamps

Perhaps the most common use of timestamps in cryptographic protocols is to counteract replay and interleaving attacks, and to provide uniqueness or timeliness guarantees. In this sense, timestamps are an alternative to challenge-response mechanisms using fresh random nonce and to message sequence numbers.

The typical operation of a system relying on timestamps is the following. Before sending a message, a party obtains a time reading from its local clock, and cryptographically binds it to the message for transmission. The receiver of the message obtains the current time from its own local clock, and subtracts the value of the timestamp received. The message is immediately rejected if the timestamp difference

is outside a fixed-size time interval, called an acceptance window, selected to account for the maximum message transit and processing time, plus clock skew between the two parties. Furthermore, the receiver keeps a list of all previously received messages whose timestamps are still within the acceptance window. The incoming message is also checked against this list, and it is rejected if it is found to be a replay.

The fact that the receiver must keep local state to detect the replay of valid time stamped messages within the acceptance window may also be considered a disadvantage. For example, a solution based on a challenge response mechanism using a random nonce could be used for the same purpose and would only require the receiver to keep a small amount of short-term state (albeit per-connection). In comparison to challenge response mechanisms, protocols using timestamps will typically require one less message to complete and will not require parties to generate random numbers. On the downside, the receiver must keep a small amount of ephemeral local state to detect the replay of valid messages within an acceptance window. The amount of state that must be kept when using timestamps can also be seen as an advantage when compared, for example, with solutions using sequence numbers where the receiver must keep static long-term state for each possible peer. In other application scenarios, there is no real alternative to the use of timestamps. Examples of this are the implementation of time-limited privileges, such as those awarded by Kerberos tickets, or the legal validity of authenticated documents, such as X.509 public key certificates.

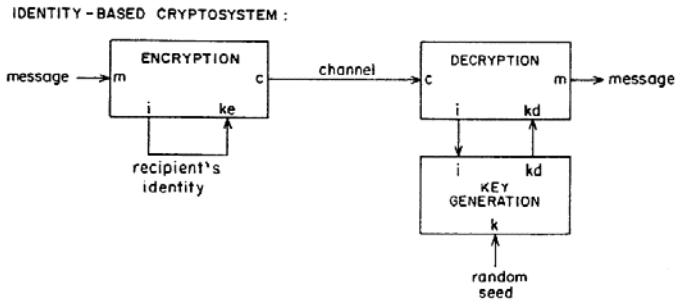
In short, timestamps are extensively used in cryptographic protocols and they are adopted in mainstream (de facto) cryptographic standards, because they have interesting security properties that can be advantageous in many real-world scenarios. However, to the best of our knowledge, the use of timestamps has not been addressed in previously published work on the theoretical security analysis of cryptographic protocols. In particular, the current formal security models for the analysis of cryptographic protocols do not allow capturing this sort of mechanism in any reasonable way. The security of this sort of mechanism relies on the use of a common time reference. This means that each party must have a local clock and that these must be synchronized to an extent that accommodates the acceptance window that is used. The local clocks must also be secure to prevent adversarial modification: if an adversary is able to reset a clock backwards, then it might be able to restore the validity of old messages; conversely, by setting a clock forward, the adversary might have advantage in preparing a message for some future point in time. These assumptions on the security and synchronization of local clocks may be seen as disadvantages of using timestamps, since in many environments they may not be realistic. For example, it is common that the synchronization of local clocks in a distributed environment is enforced by communication protocols that must themselves be secure in order for this assumption to be valid.

Organization of thesis

In this thesis the Section 1 presents an introduction on key agreement protocol and timestamps. Section two describes various ID based key agreement protocol. Section 3 describes the security attributes of key encryption protocol. Section 4 describes the proposed algorithm and the section 5 discusses the results and provides conclusion and further possibilities.

2. IDENTITY BASED KEY ENCRYPTION PROTOCOL

The basic idea behind and ID based cryptosystem is that end user can choose an arbitrary string such a email address as their public keys. This eliminates much of the overhead associated with key management.



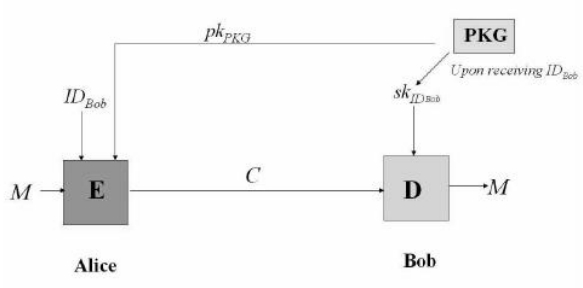
In 1984, Shamir [1] proposed a concept of identity-based cryptography. In this new paradigm of cryptography, users' identifier in-formation such as email or IP addresses instead of digital certificates can be used as public key for encryption or signature verification. As a result, identity-based cryptography significantly reduces the system complexity and the cost for establishing and managing the public key authentication framework known as Public Key Infrastructure (PKI). Although Shamir [1] easily constructed an identity-based signature (IBS) scheme using the existing RSA [5] function, he was unable to construct an identity-based encryption (IBE) scheme, which became a long-lasting open problem. Only recently in 2001, Shamir's open problem was independently solved by Boneh and Franklin [3] and Cocks [4]. Thanks to their successful realization of identity-based encryption, identity-based cryptography is now flourishing within the research community.

Basic Concepts of Identity Based Encryption and Signature

As mentioned earlier, in the IBE scheme, the sender Alice can use the receiver's identifier information which is represented by any string, such as email or IP address, even a digital image [6], to encrypt a message. The receiver Bob, having obtained a private key associated with his identifier information from the trusted third party called the Private Key

Generator (PKG)", can decrypt the cipher text. Summing up, we describe an IBE scheme us-

ing the following steps. (Figure illustrates a schematic outline of an IBE scheme).

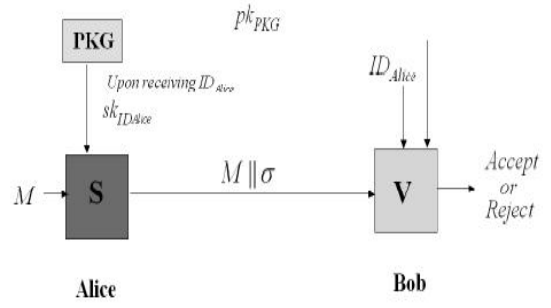


- Setup: The PKG creates its master (private) and public key pair, which we denote by skPKG and pkPKG respectively. (Note that pkPKG is given to all the interested parties and remains as a constant system parameter for a long period.)
- Private Key Extraction: The receiver Bob authenticates himself to the PKG and obtains a private key skIDBob associated with his identity IDBob.
- Encryption: Using Bob's identity IDBob and the PKG's pkPKG, the sender Alice encrypts her plaintext message M and obtains a ciphertext C.
- Decryption: Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key skIDBob to recover the plaintext M.

As a mirror image of the above identity-based encryption, one can consider an identity-based Signature (IBS) scheme. In this scheme, the signer Alice first obtains a signing (private) key associated with her identifier information from the PKG. She then signs a message using the Signing key. The verifier Bob now uses Alice's identifier information to verify Bob's signature.

{No needs for Bob to get Alice's certificate. More precisely, an IBS scheme can be described

Using the following steps. (Figure illustrates a schematic outline of an IBS scheme).



Setup: The Private Key Generator (PKG), which is a trusted third party, creates its master (private) and public key pair, which we denote by skPKG and pkPKG respectively.

Private Key Extraction: The signer Alice authenticates herself to the PKG and obtains a private key skIDAlice associated with her identity IDAlice.

Signature Generation: Using her private key skIDAlice, Alice creates a signature ¾ on her message M.

Signature Verification: Having obtained the signature ¾ and the message M from Alice, the verifier Bob checks whether ¾ is a genuine signature on M using Alice's identity IDAlice and the PKG's public key pkPKG. If it is, he returns \Accept". Otherwise, he returns \Reject".

Further studies in IDE

Smart [7] proposed an ID – Based Ak Protocol based on the idea of the IBE scheme of Boneh and Franklin. Shim [8] and Chen and Kudla [9] independently pointed out that Smart' protocol does not provide full forward secrecy – an important security requirement for key agreement protocol. Shim [8] then proposed an efficient ID based key agreement protocol that is claimed to provide full forward Secrecy, known key secrecy resilience and UK-S resilience. However it was pointed out that shim's AK protocol is vulnerable to man -in –middle attack. In 2004, Ryu et al introduced an efficient id based key agreement protocol using pairings in which computation and communication overheads for establishing a session key are significantly reduced. Wang independently showed that Ryu et al's protocol does not provide K-CI resilience .In 2005 , Lee and Lee proposed a new ID based key agreement protocol claiming that the protocol has the properties of known key secrecy (which implies key integrity), perfect forward security ,K-CI resilience and UK-S resilience . An improved version of Lee Lee Identity based key agreement protocol is presented in [10].

3. SECURITY PROPERTIES OF ID BASED KEY AGREEMENT PROTOCOL

Here we briefly provide some of the security properties of the ID based key agreement protocol. We refer the interested reader to [2] for summary properties of key agreement protocol.

Known Key Secrecy

Suppose a session key established in a session between two protocol principals (denoted by A and B) is disclosed the adversary (denoted by E) is unable to learn other session keys established in any other session . this attribute is also called key independence , which means that session keys of different sessions are computationally independent of each other . This is regarded as the standard requirement for key agreement protocols.

Key Integrity
The established session key has not been modified by an adversary or equivalently for a key agreement protocol; key integrity means that if a session key is accepted by any principal, it must be a known function of only the inputs of the protocol principals.

No Key Control
Neither A nor B can determine any portion of the shared session key established between them. Especially from the key agreement point of view , this attribute requires that the adversary E it is not able to effect the generation of the final session key . In particular, in a key control attack mounted by an adversary E, E is not able to coerce the protocol principal A and B into sharing a key when the key is not being shared with E.

Perfect Forward Secrecy
If both long term secret keys of the two protocol principal are disclosed, the adversary is unable to derive old session keys established by that two protocol principals.

Key Compromise impersonation (K-CI) resilience
Suppose A's secret key is disclosed. Obviously, an adversary who knows this secret key can impersonate A to other entities (e.g. B). However, it is desired that this disclosure does not allow the adversary to impersonate other entities (e.g. B) to A.

Unknown Key Share (UK-S) resilience
Entity A cannot be coerced into sharing a key with entity B without A's knowledge, i.e. when A believes that the key is shared with some entity C≠B and B (correctly) believes the key is shared with A.

4. PROPOSED ALGORITHM

In the proposed algorithm we use an Improved Lee Lee protocol for ID based key agreement .

Pairings
The basic definition and properties of pairings are as follows.
Let G1 be a cyclic additive group generated by an element P , whose order is prime q and G2 be a cyclic multiplicative group of the same prime q . We assume that the discrete logarithmic problem in both G1 and G2 are hard. An admissible pairings e` is a bilinear map
e` : G1*G2->G2, which satisfies the following three properties :
Bilinear : If P and Q € G1 and a ,b € Zq then e`(aP, bQ)≠e`(P,Q)^ pq.
Non Degenerate : There exists a P € G1 such that e` (P,Q)≠ 1/G2.
Computable : If P,Q € G1 , one can compute e` (P,Q)€G2 in polynomial time.
We note that modified Weil and Tate pairings associated with super singular ellipse curve are example of such admissible pairings.

Improved Lee Lee Key Agreement Protocol
The protocol involves three entities: Two users (say Alice and Bob) who wish to establish a shared secret session key and a private key generator (PKG).from whom they acquire their own private key. The ID –based key agreement protocol consists of two stages: System Setup and authenticated key agreement.

System Setup
Suppose we have an admissible pairing e` : G1×G2→ G2 as described in the preceding section where G1 and G2 are two groups with same prime order q. The PKG follows the following steps
Picks an arbitrary generator P € G1 , a secret number key s € Zq .
Choose a cryptographic hash function H!:{0,1}→ G1.
Publish the system parameters . Params = {G!,G2,e`,P,H1}.
Compute the private key SID=sQID for a user with identity information ID , in which the users public key is QID = H1(ID).
Distribute the private key SID to the user with Identity information ID via a secure channel.
Thus each user's ID based public/private key pair is defined as (QID,SID) where QID , SID € G1.

Authenticated Key agreement
We denote user Alice and Bob public/private key pairs as (Qa,Sa) and (Qb,Sb)respectively .To establish a shred session key, Alice and Bob each firstly generate an ephemeral private key(say a and b € Zq) and compute the corresponding ephemeral public keys T1a = aSA, T2A =aQB , T1B=bSB, T2B= bQA. They exchange the keys as described in figure 1.

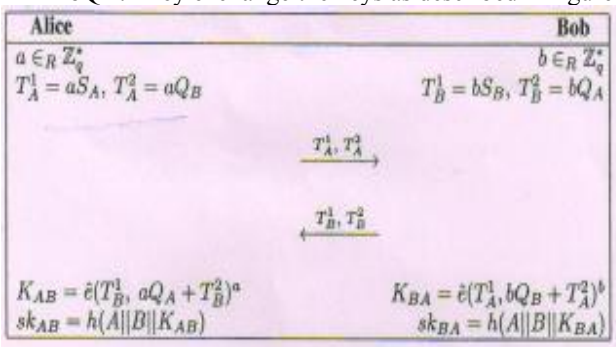


Figure 1. The Lee–Lee Protocol

After the message are exchanged the two users do the following.
Alice computes shared secret key Kab as follows
Kab = e`(T1b,aQA+T2B)^a in which T1B= bSB and T2B=bQA.
Bob computes the shared secret Kba as follows (after receiving T1A and T2A):
Kba=e`(T1A,bQB +T2A)^b
In which T1A= aSA and T2A= aQB.
Improved Lee Lee Protocol
As for this protocol the modified key generation for Alice is as follows
Sk=H(A||B||Kab||T),
In which the protocol transcript T is computed as
T=T1A||T2A||T1B||T2B

As a result if the adversary eve modifies any block that he intercepted from public channel, the two protocol transcripts calculated independently by the two protocol principals will become unequal. Hence any key control attack will become invalid.

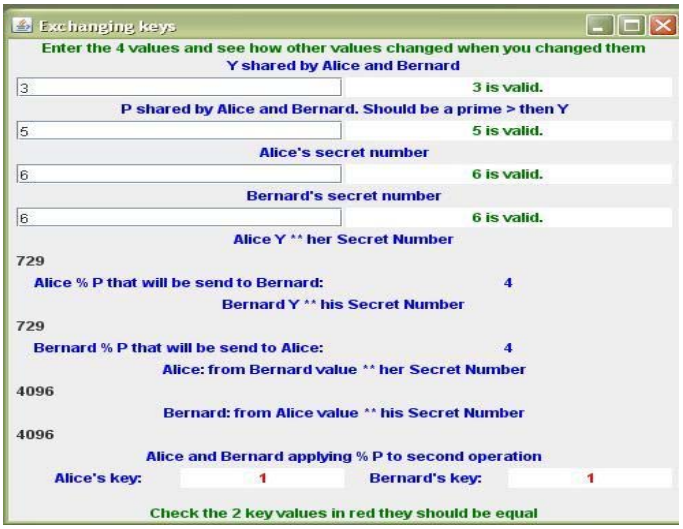
Improved Lee Lee Protocol with time stamping

The objective is to obtain a framework for the analysis of key agreement protocols relying on timestamps, where one can argue that they satisfy a formal security definition. We do not introduce an all-new time-aware analysis framework, which would mean our findings might break away from the current state-of-the-art and might not be easily comparable to previously published results. Instead, we propose to extend the existing models for the analysis of key agreement protocols in a natural way, taking care to preserve an acceptable degree of backward-compatibility. The basic idea of our approach is applicable to several competing analysis frameworks that are currently used by researchers in this area, and it does not imply the adoption of any particular one. To demonstrate this principle, we propose to extend the Lee Lee model referred above in very similar terms. The most important change that we introduce is that we provide the communicating parties with internal clocks. These clocks are the only means available to each party to determine the current (local) time. To preserve the common asynchronous trait in these models, where the adversary controls the entire sequence of events occurring during an execution, we do not allow the clocks to progress independently. Instead, we leave it to the adversary to control the individual clocks of parties: we allow it to perform a Tick (or activation) query through which it can increment the internal clock of an honest party (of course it has complete control of the clocks of corrupted parties). The adversary is not allowed to reset or cause the internal clocks to regress in any way. This restriction captures the real-world assumption we described in Section 1 that the internal clocks of honest parties must be, to some extent, secure. The addition of these elements to the Lee Lee models allows us to capture the notion of time and internal clock drifts. We preserve the asynchronous nature of the model by allowing the adversary to freely control the perception of time passing at the different parties. Through the Tick mechanism, the adversary is able to induce any conceivable pattern in the relative speed of local clocks, and may try to use this capability to obtain advantage in attacking protocols that rely on local time measurements to construct and/or validate timestamps. Of course by giving this power to the adversary, we are enabling it to drive internal clocks significantly out of synchrony with respect to each other. However, a secure protocol using explicit representations of time should make it infeasible for an adversary to take advantage of such a strategy, or at least should permit formally stating the amount of drift that can tolerated. At this point, it is important to distinguish two types of security guarantees that may be obtained from timestamps and that we aim to capture using this modelling strategy. Resistance against replay attacks. Recall that, in protocols that use timestamps to prevent replay attacks, the receiver defines an acceptance window and temporarily stores received messages until their timestamps expire. The width of the acceptance window must be defined as a trade-off between the required amount of storage space, the expected message transmission frequency, speed and processing time; and the required synchronization between the clocks of sender and receiver. Received messages are discarded if they have invalid timestamps, or if they are repeats within the acceptance window. In this setting, the local clocks are not explicitly used to keep track of elapsed time, but simply to ensure that the receiver does not have to store all previously received messages to prevent accepting duplicates . In fact, for this purpose, timestamps are essentially equivalent to sequence numbers. Furthermore, synchronization of clocks between sender and receiver is less of a timeliness issue, and more of an interoperability problem. For example, two honest parties using this mechanism might not be able to communicate at all, even without the active intervention of any adversary, should their clocks values be sufficiently apart. In our extended model, this is reminiscent of a Denial-of-Service attack, which is usually out of the scope of cryptographic security analyses. Consistently with this view and with the original models, the security definitions for cryptographic protocols using timestamps in this context remain unchanged: it is accepted that the adversary may be able to prevent successful completions of protocols (e.g. by driving internal clocks significantly out of synchronization, or simply by not delivering messages) but it should not be able to break the security requirements in any useful way. Timeliness guarantees. For protocols that use timestamps to obtain timeliness guarantees on messages, the local clock values are taken for what they really mean: time measurements. In this context, time stamped messages are typically valid for a longer period of time, and timeliness guarantees can be provided to either the sender or the receiver, or to both. For example, the sender may want to be sure that a message will not be accepted by an honest receiver outside its validity period, which is defined with respect to the sender's own internal clock. Conversely, the receiver may require assurance that an accepted message was generated recently with respect to its own local clock, where recently is quantifiable as a time interval. To deal with these guarantees we need to capture accuracy assumptions on the internal clocks of the honest parties in the system. We can do this by imposing limits on the maximum pair-wise drift that the adversary can induce between the internal clocks of different parties. In our modeling approach, we capture this sort of security requirement by stating that a protocol enforcing such a timeliness property must guarantee that any adversary breaking this requirement must be overstepping its maximum drift allowance with overwhelming probability.

5. RESULTS AND DISCUSSIONS

The above system is implemented using Java Platform. The program contains a GUI as shown in the figure below. IT

contains a key sharing program which demonstrates the key sharing protocol mentioned above.



6. CONCLUSION

In this paper we proposed a general modeling technique that can be used to extend current models for the analysis of key agreement protocols, so that they capture the use of timestamps. We have shown that popular analysis frameworks (Lee Lee) can be extended in a natural way using this technique, and that this permits addressing a new class of real-world protocols that, until now, lacked a complete formal treatment. The paper also leaves many open problems that can be addressed in future work. We conclude the paper by referring some of these topics. The approach we introduced can be applied to extend other theoretical models. Orthogonally, there are many key agreement and authentication protocols which rely on timestamps and that could benefit from a security analysis in a time-aware framework. Kerberos is an example of such a protocol, which utilizes timestamps in a setting where a server is available. In order to rigorously analyze the security of this protocol, one would need to define a timed version of three-party key agreement security models. Moving away from key agreement and authentication protocols, our approach opens the way for the formal analysis of time-related cryptographic protocols such as those aiming to provide secure message timestamping and clock-synchronization. Finally, it would be interesting to see how one could apply a similar approach to security models that try to capture public key infrastructures, where the temporal validity of certificates is usually ignored.

REFERENCES

1) Diffie, W.; Hellman, M.; , "New directions in cryptography," Information Theory, IEEE Transactions on , vol.22, no.6, pp. 644- 654, Nov 1976.
2) Boyd .C and A. Mathuria (2003) , Protocols for Authentication and key management. Springer-Verlag.
3) Dan Boneh, Matthew K. Franklin, Identity-Based Encryption from the Weil Pairing Advances in Cryptology - Proceedings of CRYPTO 2001 (2001)
4) C. Cocks, An Identity Based Encryption Scheme Based on Quadratic Residues, Cryptography and Coding - Institute of Mathematics and Its Applications International Conference on Cryptography and Coding {Proceedings of IMA 2001, LNCS 2260, pages 360{363, Springer/Verlag, 2001.
5) Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 21 (2), pages 120{126, 1978.
6) A. Sahai and B. Waters Fuzzy Identity Based Encryption, IACR ePrint Archive, Report 2004/086.
7) Smart, N.P.; , "Identity-based authenticated key agreement protocol based on Weil pairing," Electronics Letters , vol.38, no.13, pp.630-632, 20 Jun 2002.
8) Kyungah Shim; , "Efficient ID-based authenticated key agreement protocol based on Weil pairing," Electronics Letters , vol.39, no.8, pp. 653- 654, 17 April 2003.
9) Chen, L.; Kudla, C.; , "Identity based authenticated key agreement protocols from pairings," Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE , vol., no., pp. 219- 233, 30 June-2 July 2003.
10) Shuangqing Liu; Shengbao Wang; Qizhen Wang; Huan Liu; Jiajun Shen; , "Cryptanalysis and improvement of the Lee-Lee identity-based key agreement protocol," Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on , vol., no., pp.530-534, 16-18 April 2010.