One-Pass Authenticated Key Establishment Protocol on Bilinear Pairings for Wireless Sensor Networks

Manoj Ranjan Mishra[1],Jayaprakash Kar[2] and Banshidhar Majhi[3]

[1] School of Computer Application
KIIT University, Bhubaneswar, India
[2] Department of Information Systems
Faculty of Computing & Information Technology
King Abdulaziz University, Kingdom of Saudi Arabia
[3] Department of Computer Science & Engineering
National Institute of Technology, Rourkela, India *

Abstract. The article proposes one-pass authenticated key establishment protocol in random oracles for Wireless Sensor Networks. Security of the protocol relies on Computational Diffie-Hellman Problem on Bilinear Pairings. In one-pass key establishment protocol, the initiator computes a session key and a related message. The key token is to be sent to the intended receiver using receiver's public key and sender secret key. From the received key token the receiver compute the session key, which is the same as the one computed by the sender, using sender public key and receiver's secret key. Because of low communication overhead, the scheme is better suited for Wireless Sensor Networks(WSNs) than the traditional key establishment protocol to establish the session key between two adjacent nodes.

Keywords: Key establishment, bilinear pairings, Session key, Authentication, BDHP

1 Introduction

Pairing-Based Cryptography is an emerging area of cryptography that revolves around a particular function with interesting properties. Pairings, such as the Weil pairing, were first used in the context of cryptanalysis [1] to reduce the ECDLP into a discrete logarithm problem in the finite field. The first use of pairings in cryptography is the work of Sakai et al. [5] and Joux [21]. Both papers proposed pairings as the base for building complete cryptosystems. Since then many protocols have been proposed that use pairings as the underlying crypto primitives.

WSN systems are usually deployed in hostile environments where they encountered a wide variety of malicious attacks. Information that is the cooked data collected within the sensor network is valuable and should be kept confidential. In order to protect this transmitted information or messages between any two adjacent sensor nodes key establishment protocol and a mutual authentication are required for wireless sensor networks. Due to nature restrictions like low power, less storage space, low computation ability and short communication range of sensor nodes, most conventional protocols establish authenticated multiple keys between any two adjacent sensor nodes by adopting a key pre-distribution approach. However, these techniques have vulnerability. With rapid growth of cryptographic techniques, recent results show that Elliptic Curve Cryptography (ECC) is suitable for resource-limited WSNs. Cryptosystem based on Elliptic Curve Cryptography are especially interesting for sensor networks since they are more efficient in resource utilization than any other public key techniques [6] [20]. Cryptographic Protocol based on bilinear pairing is very interesting and emerging to modern cryptographic research community. Since the security of the protocol more stronger than others. Pairing based cryptography has also allowed many long-standing open problems to be solved in a well-designed way. The computational capability of sensor nodes are limited, so traditional public-key cryptography in which the computation of modular exponentiation is required, cannot be implemented on WSNs.

* Corresponding author: Jayapraksh Kar,e-mail: jayaprakashkar@yahoo.com

2 Preliminaries

2.1 Bilinear Pairings

A bilinear pairing is a map between two groups. The two kind of bilinear pairings are Weil and Tate pairings on elliptic curve. A bilinear pairing is defined as:
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q with identity element $\mathcal{O}$. Let $\mathcal{G}_T$ be a multiplicative group of order q. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive group and $\mathbb{G}_2$ is a multiplicative group. Let $\hat{e}$ be a computable and non-degenerated bilinear map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

which satisfies the following properties:

- **Bilinear**: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$, where $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$ and for $P, Q, R \in \mathbb{G}_1, \hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$.
- **Non-degenerate**: If P is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ is generator of $\mathbb{G}_2$. There exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$
- **Computability**: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

We call such a bilinear map $\hat{e}$ is an admissible bilinear pairing.

2.2 Pairing types

The properties of pairing depends on the selected groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$. Generally pairings is of three types depending three basic groups.

- `Type-1`: Here $\mathbb{G}_1 = \mathbb{G}_2$.
- `Type-2`: Pairing where $\mathbb{G}_1 \neq \mathbb{G}_2$, $\exists$ an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$.
- `Type-3`:Pairing where $\mathbb{G}_1 \neq \mathbb{G}_2$, $\nexists$ an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$

3 Security Model

In 2006 LaMacchia, Lauter & Mityagin introduced a stronger security model for AKE protocols .The model is a weak corruption model, where the adversary is allowed to reveal only the static private key of an entity through `Long-Term Key Reveal`. The protocol is modeled as a collection of programs running at different entities $e_1, e_2 \dots e_n$ with identities $ID_i, ID_j \in \{ID_1, ID_2 \dots ID_n\}$. Each entity is allowed to have different instances of running the protocol, modeling the real time scenario of having multiple sessions open with different partners. The adversary $\mathcal{A}$ performs polynomial bounded number of queries to the oracles provided to $\mathcal{A}$ by $\mathcal{B}$. Let us assume that, $\mathcal{A}$ can activate each user not more than m number of times. Challenger $\mathcal{B}$ runs $\mathcal{A}$ as subroutine and selects $ID_i, ID_j \in \{ID_1, ID_2 \dots ID_n\}$ and $s \in \{1, 2 \dots m\}$ *i.e* oracle $\{\prod_{i,j}^{s}\}$, where $\prod_{i,j}^{s}$ behaves as node i carrying out the protocol with node j for the s times. If $\mathcal{B}$ selects the s^th session activated at ID_i with ID_j as the test session, then $\mathcal{B}$ guess will be correct. $\mathcal{B}$ sets the public key of KGC as aP. The communications network is controlled by a PPT adversary $\mathcal{A}$, which schedules and mediates all sessions between the entities. It is also given the power of initiating fictitious entities by obtaining private keys from the KGC for arbitrary identities. All the entities (including the honest ones) are activated by $\mathcal{A}$. Upon activation, the entities perform some computations as per the received communication, update their internal state and complete the session. The session identifier *sid* is assumed to be the concatenation of the messages exchanged between two entities along with their identities. Two sessions are said to be matching sessions if their *sid*'s are identical. Let msk be the master secret key used by the KGC to issue private keys to the users. $\mathcal{A}$ selects identities of the honest entities and let them obtain private keys for the identities from the KGC. It is allowed to obtain private keys for any arbitrary identity of its choice. $\mathcal{A}$ run the following queries:

- `Send`: In this query, a unique message m is to be send to ID_u that comes from ID_v. The result will submit to $\mathcal{A}$. If m is null message($i.e\ m = \phi$), the queries active the entity ID_u as `initiator` otherwise its role is considered as `responder`. Thus the session sid is defined as the tuples $<ID_u, ID_v, m, \texttt{role}>$, where $\texttt{role} \in \{\texttt{initiator}, \texttt{responder}\}$
- `Reveal(`sid`)`: The session key sid generated by the challenger $\mathcal{B}$ in the session sid returns to $\mathcal{A}$.
- `Long-term Key Reveal(`ID_u`)`: In this query, the challenger $\mathcal{B}$ takes ID_u and returns the `Long-term Private key`.
- `Ephemeral Key Reveal(`sid`)`: The challenger $\mathcal{B}$ returns the ephemeral private key used in the incomplete session sid to $\mathcal{A}$. This may includes all sensitive session state information used by the entity ID_u in sid. It is assumed that, when the session is completed, the session state information is erased from the memory of the entity.
- `Extract(`ID_u`)`: The challenger $\mathcal{B}$ takes ID_u returns the private key to $\mathcal{A}$.

The adversary $\mathcal{A}$ issues a `Test` query to a clean session sid as follows:

1. `Test(`sid`)`: Here the adversary $\mathcal{A}$ begins by running the oracle $\prod_{u,v}^{s}$ and asks a single new query `Test` on the session sid. To answer the query the oracle flips a fair coin $b \in \{0, 1\}$ and returns the session key in the session sid to $\mathcal{A}$ if $= 0$ or a random value if $b = 1$. Finally it terminates the game if he guesses the correct value b' which is different from the random value. The adversary $\mathcal{A}$ wins the game if $b' = b$ and selected test session is clean.
2. `clean(`sid`)`: Let sid be a session completed at the entity ID_u and let sid^* be the matching session at ID_v. The long-term secret keys ID_u and ID_v are denoted by SK_u and SK_v respectively. Let E_{SK_u} and E_{SK_v} be the ephemeral secret keys used in sid and sid^* respectively. A session $sid$is said not to be clean if one of the following conditions holds:
   - ID_u or ID_v is an adversary controlled party (whose private keys are obtained by `Extract` queries).
   - $\mathcal{A}$ reveals the master secret key msk of the KGC
   - $\mathcal{A}$ reveals the session key held in sid or sid^* (if there exists such a sid^*).
   - A matching session sid^* exists and $\mathcal{A}$ reveals SK_u and E_{SK_u} or SK_v and E_{SK_v}.
   - A matching session sid does not exist and $\mathcal{A}$ reveals SK_u and E_{SK_u} or SK_v.

   In all other cases the session is said to be `clean`.

   Definition 1. *The advantage of the adversary $\mathcal{A}$ in the experiment of the protocol π is defined as*

$$Adv_{\pi}(\mathcal{A}) = Pr[Suc] - \tfrac{1}{2}$$

   An one-pass authenticated protocol pi is said to be secure under the defined model, if there does not exist PPT adversary that has no negligible advantage to distinguish the real session key from a random bit string. i.e

$$Adv_{\pi}(\mathcal{A}) = \epsilon(k) \leq k^{-c}\ \forall\ k > k_c, c > 0$$

4 Proposed One-Pass Protocol

Consider two arbitrary nodes i and j would like to share session keys to establish secure communication. Node i has computed long-term private and public key as $S_i = s \cdot Q_i$. Similarly node j has computed his long-term private and public key as $S_j = \lambda_j \cdot Q_j$. The protocol comprises the following three polynomial time solvable (PPT) algorithms.

- **Setup** : Given security parameters k, the KGC chooses groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order q. A generator P of $\mathbb{G}$, a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and collision resistant hash function $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}$, $\mathcal{H}_2 : \mathbb{G} \times \{0,1,\}^* \to \mathbb{F}_q^*$, $\mathcal{H}_3 : \mathbb{G} \to \mathbb{F}_q^*$. A key derivation function $\mathcal{F} : \mathbb{G}_T \to \{0,1\}^k$. Where $|k|$ is the length of the key strings. It chooses a master-key $s \in \mathbb{F}_q^*$ and computes $P_{pub} = sP$. The KGC publishes the system public parameters $params = <\mathbb{G}, \mathbb{G}_T, \hat{e}, q, p, P_{pub}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3>$.
- **Extract** : KGC takes the input the identities ID_i and ID_j of node i and j respectively and runs the algorithm and Computes the private keys as $S_i = s \cdot Q_i$ and $S_j = s \cdot Q_j$ of the node i and j respectively. (Where Public key of node i and j are Q_i and Q_j respectively. $Q_i = \mathcal{H}_1(ID_i)$ and $Q_j = \mathcal{H}_1(ID_j)$.

– **Protocol**: The protocol involves by considering two arbitrary nodes i and j. Follows the following steps
  - Node i chooses randomly $\lambda \in \mathbb{Z}_q^*$ and computes $U = \lambda \cdot Q_i$.
  - Computes $\mathcal{H}_2(U, ID_i \| ID_j) = \Gamma$ and $h = \mathcal{H}_3(\Gamma)$.
  - Computes $SK_{ij} = \hat{e}((\lambda + h)S_i, Q_j)$
  - Computes $V = \lambda \cdot \Gamma$.
  - Sends the pair of elements (U, V) to **Node** j.
  - After received, Node j computes $\Gamma = \mathcal{H}_2(U, ID_i \| ID_j)$, $h = \mathcal{H}_3(\Gamma)$ and $W = s \cdot \Gamma$.
  - Verify $\hat{e}(U, W) \stackrel{?}{=} \hat{e}(S_i, V)$ and computes $SK_{ji} = \hat{e}(U + hQ_i, S_j)$

4.1 Proof of Correctness

$$\begin{aligned}\hat{e}((\lambda + h)S_i, Q_j) &= \hat{e}(S_i, Q_j)^{(\lambda+h)} \\ &= \hat{e}(Q_i, Q_j)^{s(\lambda+h)} \\ \hat{e}(U + hQ_i, S_j) &= \hat{e}(\lambda Q_i + hQ_i, S_j) \\ &= \hat{e}(Q_i, S_j)^{(\lambda+h)} \\ &= \hat{e}(Q_i, Q_j)^{s(\lambda+h)}\end{aligned}$$

Consider the verification equation $\hat{e}(U, W) = \hat{e}(S_i, V)$ and check the equality as

$$\begin{aligned}\hat{e}(U, W) &= \hat{e}(\lambda Q_i, sQ) \\ &= \hat{e}(Q_i, Q)^{\lambda s} \\ \hat{e}(S_i, V) &= \hat{e}(Q_i, Q)^{\lambda s}.\end{aligned}$$

1. **Setup** : Given security parameters k, the KGC chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q.
   A generator P of $\mathbb{G}$, a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and collision resistant hash function $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}$, $\mathcal{H}_2 : \mathbb{G} \times \{0,1,\}^* \to \mathbb{F}_q^*$, $\mathcal{H}_3 : \mathbb{G} \to \mathbb{F}_q^*$.
   It chooses a master-key $s \in \mathbb{F}_q^*$ and computes $P_{pub} = sP$.
   The KGC publishes the system public parameters $params = <\mathbb{G}, \mathbb{G}_T, \hat{e}, q, p, P_{pub}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3>$.
2. **Extract** : KGC takes the input the identities ID_i and ID_j of node i and j respectively and runs the algorithm and Computes the private keys as $S_i = s \cdot Q_i$ and $S_j = s \cdot Q_j$ of the node i and j respectively. $Q_i = \mathcal{H}_1(ID_i)$ and $Q_j = \mathcal{H}_1(ID_j)$.
   (Where Public key of node i and j are Q_i and Q_j respectively.
3. **Protocol**: The protocol involves by considering two arbitrary nodes i and j

Node i		**Node** j
Chooses randomly $\lambda \in \mathbb{Z}_q^*$ and computes $U = \lambda \cdot Q_i$.		
Computes $\mathcal{H}_2(U, ID_i \| ID_j) = \Gamma$ and $h = \mathcal{H}_3(\Gamma)$.		
$SK_{ij} = \hat{e}((\lambda + h)S_i, Q_j)$		
Computes $V = \lambda \cdot \Gamma$.		
	(U, V) $\longrightarrow$	
		Computes $\Gamma = \mathcal{H}_2(U, ID_i \| ID_j)$, $h = \mathcal{H}_3(Q)$
		Computes $W = s \cdot \Gamma$
		and Verify $\hat{e}(U, W) \stackrel{?}{=} \hat{(}S_i, V)$.
		Computes $SK_{ji} = \hat{e}(U + hQ_i, S_j)$

Fig. 1. Proposed One-Pass Protocol

5 Security Analysis

This section describes the security analysis and prove that the scheme is provably secure win random oracle model.

Theorem 1 *In the random oracle model, the proposed one-pass protocol is secure, if the adversary $\mathcal{A}$ can computes and distinguish the real session key from a given random oracle during the game with a non-negligible advantage and run* `Keygen` *queries,* `Send`, `Reveal`, `Extract`, `Ephemeral Key Reveal` *and* `Test` *queries; then there exists a challenger $\mathcal{B}$ that can solve an instances of Bilinear Diffie-Hellman problem with a non-negligible advantage.*

Proof:

– `Setup`: The Challenger $\mathcal{B}$ receives a random instance $(P, aP, bP, cP$ of the Bilinear Diffie-Hellman problem. His goal is to compute $\hat{e}(P,P)^{abc}$. $\mathcal{B}$ will run $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the game. $\mathcal{B}$ needs to maintain lists L_1, L_2 and L_3 that are initial empty and are used to keep track of answers to queries asked by $\mathcal{A}$ to oracles $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ respectively. Consider there are n number of users with identities $ID_1, ID_2 \ldots ID_n$ are participating in the protocol. Let us assume that, $\mathcal{A}$ can activate each user not more than m number of times. $\mathcal{B}$ selects $ID_i, ID_j \in \{ID_1, ID_2 \ldots ID_n\}$ and $s \in \{1, 2 \ldots m\}$ *i.e* oracle $\{\prod_{i,j}^{s}\}$, where $\prod_{i,j}^{s}$ behaves as node i carrying out the protocol with node j for the s times. If $\mathcal{B}$ selects the s^th session activated at ID_i with ID_j as the test session, then $\mathcal{B}$ guess will be correct. $\mathcal{B}$ sets the public key of KGC as aP. $\mathcal{B}$ run the queries of $\mathcal{A}$ and answers the followings:

Oracle Simulation:

1. $\mathcal{H}_1$**-Oracle**: At the beginning of the game, For $\mathcal{H}_1$-queries the list L_1 is empty. $\mathcal{B}$ gives $\mathcal{A}$ the system parameters with $P_{pub} = cP$ (c is unknown to $\mathcal{B}$ and plays the role of the KGC's master-key). Then $\mathcal{B}$ chooses two distinct random numbers $i, j \in \{1 \ldots q_{\mathcal{H}_1}\}$. $\mathcal{A}$ asks a polynomial bounded number of $\mathcal{H}_1$ requests on identities of his choice. At the i^{th} $\mathcal{H}_1$ request, $\mathcal{B}$ answers by $\mathcal{H}_1(ID_i) = aP$. At the j^{th}, he answers by $\mathcal{H}_1(ID_j) = bP$. Since aP and bP belong to a random instance of the BDH problem, $\mathcal{A}$'s view will not be modified by these changes. Hence, the private keys S_{ID_i} and S_{ID_j} (which are not computable by $\mathcal{B}$) are respectively acP and bcP. Thus the solution $\hat{e}(P,P)^{abc}$ of the BDH problem is given by $\hat{e}(Q_{ID_i}, S_{ID_j}) = \hat{e}(S_{ID_i}, Q_{ID_j})$. For requests $\mathcal{H}_1(ID_k)$ with $k \neq i, j$, $\mathcal{B}$ chooses $b_k \leftarrow_R \mathbb{F}_q^*$, adds the pair (ID_k, b_k) in list L_1 and answers $\mathcal{H}_1(ID_k) = b_k P$. Further on input $ID_i \in \{0,1\}^*$, $\mathcal{B}$ first checks the L_1-list $<ID_i, X_i, q_i, x_i$, if $ID_i = ID_B>$, selects new random $\gamma_i \leftarrow_R \mathbb{F}_q^*$, sets $X_i = b \cdot P, q_i = \gamma_i$, add this tuple $<ID_i, X_i, q_i, *>$ to the L_0-list and returns q_i. Otherwise, $\mathcal{B}$ selects a new random $\gamma_i \leftarrow_R \mathbb{F}_q^*$, $x_i \leftarrow_R \mathbb{F}_q^*$, sets $X_i = x_i \cdot P, q_i = \gamma_i$, add this tuple $<ID_i, X_i, q_i, x_i>$ to the L_1-list and returns q_i. $\mathcal{B}$ starts with empty list L_1. It takes the input ID_u and checks if there is entry for it in L_1. If exists, then returns the stored Q_u to $\mathcal{A}$. Otherwise, it chooses a value $\psi_u \in \mathbb{F}_q^*$ randomly and computes $\psi_u \cdot P$. It returns it to $\mathcal{A}$. The entry $<ID_u, \psi_u, Q_u>$ includes in the list L_1. If $u = j$, the value bP from its input is returned and entry $<ID_u, \perp, bP>$ is added to the list.
2. $\mathcal{H}_2$**-Oracle**: On input $(U, ID_i \| ID_j)$, $\mathcal{B}$ first checks the L_2-List, whether the tuple $<ID_i, ID_j, U, Q>$ in the L_2-List, $\mathcal{B}$ returns Γ, otherwise $\mathcal{B}$ chooses a new random $\Gamma \in \mathbb{G}$, adds Γ to the L_2-list and return Γ.
3. $\mathcal{H}_3$**-Oracle**: On input the element Γ, $\mathcal{B}$ first checks the L_3-List, whether the tuple $<\Gamma, h>$ in the L_3-List, $\mathcal{B}$ returns h, otherwise $\mathcal{B}$ chooses a new random $h \leftarrow_R \mathbb{F}_q^*$, includes h to the L_3-list and return h.
4. `Keygen-Oracle`: When $\mathcal{A}$ makes a $Keygen$ query with ID_i as the input, $\mathcal{B}$ checks the L_1-List to verify whether or not there is an entry for ID_i. If the L_1-List does not contain an entry for ID_i, return $\perp$. Otherwise, if $ID_i = ID_B$, $\mathcal{B}$ recovers the tuple $<ID_i, X_i, q_i, x_i>$ from the L_0-List and returns $<X_i, q_i, *, *>$, if $ID_i \neq \{ID_i\}_{i=1 \ldots n}$, $\mathcal{B}$ recovers the tuple $<ID_i, X_i, q_i, x_i>$ from the L_1-List and returns $<X_i, q_i, S_{ID_i}, d_i>$, where $S_{ID_i} = x_i(aP) = a(x_i P) = aX_i$ and $d_i \leftarrow_R \mathbb{F}_q^*$ is randomly selected.
5. `Send-Oracle`: $\mathcal{A}$ makes a $Send(ID_u, ID_v, m)$ query base on the observation to the session *sid*:
   (a) The node u with ID_u originates $m = \mu$. If $u = 1, v = j$ with s^{th} session between them, $\mathcal{B}$ randomly selects $\gamma \in \mathbb{F}_q^*$ and returns $X_s = \lambda\gamma \cdot (cP)$. If $u = i$, $\mathcal{B}$ chooses $\eta_t, h_{j,t} \in \mathbb{F}_q^*$ computes $X_{j,t} = \eta_t P - h_{j,t} Q_j$ and returns $h_{j,t}$ by running $\mathcal{H}_2$ corresponds to the query $(X_{j,t}, ID_j \| ID_v)$, $\mathcal{B}$ stores η_t in the list L_2 for session *sid*. Similarly for other session between ID_i and ID_j, $\mathcal{B}$ chooses
   (b) The role of ID_i is responder ($m \neq \mu$). $\mathcal{B}$ accepts the session and marks it as completed.
6. `Reveal(`*sid*`)-Query`: Let the s-session between the node ID_i and ID_j, $\mathcal{B}$ aborts the simulation. Let the session is at ID_j, $h_{j,t} = \mathcal{H}_2(X_{j,t}, ID_j \| ID_v)$. $\mathcal{B}$ retrieves

η_t and returns $\mathcal{F}(\hat{e}(\eta_t(aP, Q_v)) = \mathcal{F}(\hat{e}(\eta_t P - h_{j,t}Q_j + h_{j,t}Q_j, Q_v))^a = \mathcal{F}(\hat{e}(X_{j,t} + h_{j,t}Q_j, Q_v))^a$. In other cases the session key with its knowledge of static and ephemeral private keys.

7. `Extract/Long-Term Key Reveal(`*sid*`)-Query`: $\mathcal{B}$ first checks to see if there is an entry corresponding to ID_u in L_1. On no match, it makes a $\mathcal{H}_1$ query with the input ID_u. It retrieves the value ψ_u from L_1 and returns $\psi_u(xP)$. Note that an `Extract` or `Long- Term Key Reveal query` on the input ID_j does not have to be simulated by $\mathcal{B}$. On such a query $\mathcal{B}$ aborts its execution.
8. `Ephemeral Key reveal-Query`: If this is the s^{th} session between ID_i and ID_j, $\mathcal{B}$ outputs fail. For all the sessions at ID_j, the `Ephemeral Key Reveal` is not handled. In all other cases, $\mathcal{B}$ returns the corresponding ephemeral private key it has chosen while answering the `Send` queries.
9. `Test(`*sid*`)-Query`:If this is not the s^{th} session between ID_i and ID_j, $\mathcal{B}$ outputs fail. *sid* is the anticipated session but if it is not clean then $\mathcal{B}$ aborts its simulation. Otherwise, $\mathcal{B}$ has to return the session key held in the session *sid* or a random value from the session key distribution after tossing a coin. However, as shown below $\mathcal{B}$ cannot compute the real session key, which would be of form:

$SK = \mathcal{F}(\hat{e}(Z_{i,s} + h_{i,s}Q_i, Q_j)^a)$
$= \mathcal{F}(\hat{e}(\lambda_i\gamma(cP) + h_{i,s}\lambda_i P, bP)^a)$
$= \mathcal{F}(\hat{e}((\gamma c + h_{i,s})\lambda_i P, bP)^a)$
$= \mathcal{F}(\hat{e}(aP, bP)^{(\gamma c + h_{i,s})\lambda_i}$
It is required to solve BDH problem with instances $<aP, bP, c\lambda_i\gamma P>$.

Table 1. Security Comparison

*Scheme	IKA	PFS	KKS	UKS	KcI	LoI
Yuan Wang(Protocol-I) [15]	√	√	√	×	×	×
Yuan Wang(Protocol-II) [15]	√	×	√	×	×	×
Gorantla *et al.* [4]	√	√	√	×	√	×
Konstantinos Chalkias *et al.* [14]	√	×	√	×	×	×
Proposed Protocol	√	√	√	√	√	√

* IKA : Implicit Key Authentication, PFS : Perfect forward secrecy
KKS : Known Key Secrecy, UKS : Unknown Key Share
KcI : Key-compromise Impersonation, LoI : Loss of Information

Table 2. Comparison of Computational Cost

	Sender		*Receiver*	
	$Pairing$	$Mul(\mathbb{G})$	$Pairing$	$Mul(\mathbb{G})$
Yuan Wang(Protocol-I) [15]	2	1	2	1
Yuan Wang(Protocol-II) [15]	2	1	2	2
Gorantla *et al.* [4]	1	2	1	1
Proposed Protocol	1	2	1	1

6 Implementation issues

We can follow the similar technique used in [13] to implement the protocol in the single-hop setting in which each sensor node can establish the session key. We assume that the system public parameters and the master secret key are generated by the base station and embedded on each sensor node during the deployment. We assume that the base station is

powerful enough to perform computationally intensive cryptographic operations like pairing, hashing etc , and the sensor nodes, on the other hand, have limited resources in terms of computation, memory and battery power. We can use the sensor nodes MicaZ 3 in the implementation . The nodes are developed by Crossbow Technology. Its RF transceiver complies with IEEE 802.15.4/ZigBee, and the 8-bit microcontroller is Atmel ATmega128L, a major energy consumer. We used a PC of latest configuration as a base station. The program can be developed in nesC, C and Java. The base operating system for the MicaZ platform is TinyOS 2.0.

Let us consider n no of sensor nodes as $sn_1, sn_2 \dots sn_n$ with identity $ID_1, ID_2 \dots ID_n$. The system parameters $(\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{pub}, \mathcal{H}_0, \mathcal{H}_1)$ is generated by the base station and all parameters will be embedded on each sensor node. One of the node will computes (U, V) and sends to its adjacent nodes. After receiving, the nodes will pass though the verification equation described in the protocol and computes the session key.

6.1 Pairing Algorithms

The most efficiently computable pairings are Weil and Tate pairing on elliptic curve. Tate pairing is more efficient than the Weil pairing. We consider `Type-1` pairing on super singular elliptic curve in our proposed protocol. Tate pairing is the bilinear mapping $\hat{e}(P, Q)$, where P and Q are two arbitrary linearly independent points on an elliptic curve $\mathbb{E}(\mathbb{F}_q^k)$, evaluates as an element of extension field $\mathbb{F}_{q^k}$. If P is of prime order r, then the pairing is evaluated as an order of r. We can apply algorithm-1 to compute the Tate pairing for implementation in more efficient way in term of memory space, bandwidth and proceesing speed.

We consider the following elements for implementation of pairing

- As we have discussed `Type-1` pairings is more suitable on super singular curves, these curves can be divided into three sub-classes curves over binary fields as $q = 2^m$ with $k = 4$, curves over field of large prime characteristics 3 $i.e$ $q = 3^m$ with $k = 6$ and curves over field of large prime characteristics $q = p, p > 3$ with $k = 2$. The most suitable curve for implementation on 8-bit processor is curves over filed of prime characteristics is $k = 4$.
- The binary field $\mathcal{F}_{2^271}$ can be chosen to achieve the security.
- Super singular curve is

$$y^2 + y = x^3 + x \tag{1}$$

  The number of points on the curves is $2^{271} + 2^{136} + 1 = 487805.r$. Where r is a large prime.

The following table summarizes the cost of $\eta_T(P, Q)$ on $y^2 + y = x^3 + x$.

Table 3. cost of $\eta_T(P, Q)$ on $y^2 + y = x^3 + x$

Execution	$\|Mul\|$	$\|Sqrs\|$	$\|Sqrt\|$
Main loop	1904	544	544
Final loop	114	139	0

7 Conclusion

Here we have proposed a novel construction of one-pass key establishment protocol for WSNs which have the memory space required for each node is fixed. Also here the sensor node can establish secure communications with other adjacent nodes. The protocol is secure against perfect forward key secrecy and modification attack. Security of the proposed protocol relays on Bilinear Diffie-Hellman Problem. It achieves the security goals `Known Key secrecy`,`perfect forward secrecy`, `unknown key share`, `key compromise impersonation` and `key control`.

Algorithm 1: Computation of $\eta_T(P,Q)$ on $y^2+y=\ x^3+x+b$ curve over $\mathbb{F}_{2^m}$

Input P, Q
Output $\eta_T(P,Q)$
$P \leftarrow (x_P, y_P), Q = \ (x_Q, y_Q)$
$\theta \leftarrow \ x_P + 1$
$\sigma_1 \leftarrow \theta \cdot (x_P + x_Q + 1) + y_P + y_Q + 1 + (\theta + x_Q)u + v$
for $i = 1$ to $(m+1)/2$ **do**
  $\theta \leftarrow x_P, x_P \leftarrow \sqrt{x_P}, y_P \leftarrow \sqrt{y_P}$
  $\sigma_2 = \ \theta \cdot (x_P + x_Q) + y_P + y_Q + x_P + (\theta + x_Q)u + v$
  $\sigma_1 \leftarrow \sigma_1 \cdot \sigma_2$
  $x_Q \leftarrow x_Q^2$, $y_Q \leftarrow y_Q^2$
end for
return $\sigma_1^{(2^{2m}-1)(2^m-2^{(m+1)/2+1})}$

References

1. A. Menzes, T. Okamoto, and S. Vanstone Reducing elliptic curve logarithms to logarithms in a finite field, IEEE Transactions on Information Theory, 39, pp. 1639–1646, 1993.
2. H. Krawczyk HMQV: A high-performance secure Diffie-Hellman protocol, in Proceeding of Advances in Cryptology - Crypto 05, LNCS 3621, pp. 546-566, Springer-Verlag, 2005.
3. K. Lauter, and A. Mityagin Security Analysis of KEA Authenticated Key Exchange Protocol, in Proceeding of Public Key Cryptography - PKC-06, LNCS 3958, pp. 378-394, Springer Verlag, 2006.
4. M.Choudury Gorantla, Colin Boyd and Juan Manuel Gonzalez Nieto ID-based One-pass Authenticated Key Establishment, in proceeding of Australian Information Security Conference (AISC2008), Wollongong, Australia, January 2008. Con- ferences in Research and Practice in Information Technology (CRPIT), Vol. 81, 2008.
5. R. Sakai, K. Ohgishi, and M. Kasahara Cryptosystems based on pairing. Symposium on Cryptography and Information Security, Okinawa, Japan,2000.
6. D. Hankerson, A. Menezes, and S. Vanstone Guide to Elliptic Curve Cryptography, Springer, 2004.
7. V.S. Miller Use of elliptic curves in cryptography, in: Proceedings of the Advances in Cryptology - Crypto'85, New York, USA, 1985, pp. 417-426.
8. M. Bellare and P. Rogaway Entity Authentication and Key Distribution. In proceedings of Crypto 1993, LNCS 773, pp. 231-249, Springer-Verlag, 1994.
9. M. Bellare and P. Rogaway Provably Secure Session Key Distribution: The Three-party Case. In proceedings of STOC 1995, pp. 57-66, ACM Press, 1995.
10. M. Bellare, D. Pointcheval, and P. Rogaway Authenticated Key Exchange Secure Against Dictionary Attacks. In proceedings of Eurocrypt 2000, LNCS 1807, pp. 139-155, Springer-Verlag, 2000
11. N. Koblitz Elliptic curve cryptosystem, Mathematics of Computation 48 (1987), 203-209.
12. N. Koblitz. *A course in Number Theory and Cryptography ,2nd edition* Springer-Verlag-1994
13. Joseph K. Liu, Joonsang Baek, Jianying Zhou, Yanjiang Yang and Jun Wen Wong *Efficient Online/Offline Identity-Based Signature for Wireless Sensor Network*, in IACR Arcieve ePrint-2010/03.
14. Konstantinos Chalkias, Spyros T. Halkidis, Dimitrios Hristu-Varsakelis, George Stephanides, and Anastasios Alexiadis A Provably Secure One-Pass Two-Party Key Establishment Protocol, Information Security and Cryptology, Lecture Notes in Computer Science Volume 4990, 2008, pp 108-122, 2008
15. Yuan Wang, Duncan S. Wong, and Liusheng Huang One-Pass Key Establishment Model and Protocols for Wireless Roaming with User Anonymity, International Journal of Network Security, Vol.16, No.2, PP.129-142, Mar. 2014
16. K. H Rosen *"Elementary Number Theory in Science and Communication"*, 2nd ed., Springer-Verlag, Berlin, 1986.
17. A. Menezes, P. C Van Oorschot and S. A Vanstone *Handbook of applied cryptography. CRC Press, 1997.*
18. J.Kar and B.Majhi An Efficient Password Security of Three Party Key Exchange Protocol based on ECDLP" 12th International Conference on Information Technology 2009 (ICIT 2009), Bhubaneswar, India, Tata McGrow Hill Education Private Limited, pp75-78, 2009.
19. J.Kar and B.Majhi An Efficient Password Security of Multiparty Key Exchange Protocol based on ECDLP" International Journal of Computer Science and Security (IJCSS) , Malyasia, Vol.3 (5), pp 405-413, Nov 2009.

20. J.Kar and B. Majhi A Secure Two-Party Identity Based Key Exchange Protocol based on Elliptic Curve Discrete Logarithm Problem", Journal of Information Assurance and Security, USA Vol-5(1), pp 473-482, 2009.
21. A. Joux A one round protocol for tripartite Diffie-Hellman., Journal of Cryptology, 17 (2004), pp. 263–276. Proceedings of ANTS-IV, 2000.
22. Wang.Y Efficient Identity-Based and Authenticated Key Agreement Protocol, Cryptology ePrint Archive, Report 2005/108.
23. Aumann Y.and Rabin M. Authentication, enhanced security and error correcting codes,Advances in Cryptology - Crypto'98, LNCS, 1462, 299-303.