Journal of Information Security and Applications 35 (2017) 44–50

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Design and analysis of pairing protocol for bluetooth enabled devices using R-LWE Lattice-based cryptography

Samta Gajbhiye[a,*], Sanjeev Karmakar[b], Monisha Sharma[c], Sanjay Sharma[d]

[a] Computer Science and Engineering Department, FET, SSGI, SSTC, CSVTU, Bhilai, Chhattisgarh, India
[b] Master of Computer Application, BIT, CSVTU, Bhilai, Chhattisgarh, India
[c] Electronics and Telecommunication Department, FET, SSGI, SSTC, CSVTU, Bhilai, Chhattisgarh, India
[d] Department of Mathematics, BIT, CSVTU, Bhilai, Chhattisgarh, India

ARTICLE INFO

Article history:

Key words:
Secure Simple Pairing
Elliptic curve Diffie–Hellman
Numeric comparison
Quantum computers
Lattice-based cryptography
Ring-Learning with Errors

ABSTRACT

Secure Simple Pairing protocol of Bluetooth introduced in Bluetooth version 2.1+EDR has security issues. Shared secret key using Elliptic Curve Diffie–Hellman algorithm generated in public key exchange phase, is subsequently used for authentication of devices and for computing Link key. The stability of Elliptic-Curve-Diffie–Hellman is based on the determination of order and structure of a finite Abelian group and computation of Elliptic Curve Discrete Logarithm (ECDLP) in a cyclic group. An algorithm for a quantum computers exists, that can figure out the inverse of Elliptic Curve Discrete Logarithm in polynomial time. Therefore, it demands a safe and sound cryptosystem for quantum computers. A revision to boost the security of pairing and authentication process in the Secure Simple Pairing (SSP) protocol of Bluetooth intended for classical and quantum computers is proposed called as Classiquantum Resistance Secure Simple Pairing protocol. On simulating the proposed protocol, it is studied that the proposed protocol is superior over SSP, as far as security performance by enhancement of key strength and by wrapping link key with two R-LWE secret key, is concerned.

1. Introduction

Up to Bluetooth 2.0+EDR, symmetric key cryptography was a base for pairing, where both devices shared the identical PIN (Personal Identification Number) code [5]. Later in Bluetooth 2.1+EDR, a fresh pairing mechanism named Secure Simple Pairing (SSP) is introduced, intended to improve security in piconet and scatter net [6]. Unlike Bluetooth 2.0+EDR, SSP in Bluetooth 2.1+EDR exploits Elliptic Curve Diffie–Hellman (ECDH) public-key cryptography in its pairing protocol. Ever since, during pairing researchers observed and introduced many attacks. In addition, researchers too utilized elliptic curve method to improve the Bluetooth pairing algorithm. In today's scenario, standard and classical computers employing elliptic curve cryptosystems are secure, because the inverse of the discrete logarithm problem on elliptic curves is liable to be unbreakable [19] than the traditional Discrete Logarithm Problem (DLP) [8]. Nevertheless, the same is not true for future quantum computers [26].

Lattice-based cryptographic constructions hold a great promise for post-quantum cryptography, as they enjoy very strong security proofs based on worst-case hardness [2,15]. Thus, a quantum resistance pairing protocol for quantum computers using Lattice-Based Ring Learning With Errors (R-LWE), named as Classiquantum Resistance Secure Simple Pairing (CRSSP) is proposed over existing SSP. Implementing enhanced security measures, such as those, which have been proposed here, would reduce the security risks of current pairing protocol.

This paper is organized as follows in sections wherein, in Section 2, SSP with numeric comparison mode and Lattice-based cryptography for quantum computers is discussed. Proposed Bluetooth pairing protocol CRSSP and its advantages over SSP are discussed in Section 3 and 4 respectively followed by results and discussion in Section 5. Finally, concluded the paper in Section 6.

2. Secure Simple Pairing in bluetooth and attacks

2.1.1. Secure Simple Pairing in bluetooth with numeric comparison mode

After the unencrypted exchange of IO capability and device address, each device generates its own ECDH public-private key pair and computes the Diffie–Hellman shared secret key (Eq. (1)). In authentication Stage 1, with numeric comparison mode, nonce are exchanged and then both devices, to validate each other (Eq. (2))

* Corresponding author.
E-mail address: samta.gajbhiye@gmail.com (S. Gajbhiye).

http://dx.doi.org/10.1016/j.jisa.2017.05.003
2214-2126/

use this nonce. Stage 2 of authentication then certifies that both devices have efficiently completed the exchange using Challenge-Response scheme (Eq. (3)). On successful completion of authentication Stage 2, both the parties compute link key (Eq. (4)) with parameters DHKey, Nonce, public key, Bluetooth address and IO capability. This link key is further used to generate an encryption key for encryption of messages or file. Here, $g(\ldots)$ $f_3(\ldots)$ and $f_4(\ldots)$ are HMAC-SHA256 functions and $f_1(\ldots)$ is Diffie–Hellman algorithm [6].

Phase 2: Public Key Exchange

$$SecretKey = f_1(ECPrivateKey, ECPublicKeyAlice, ECPublicKeyBob) \quad (1)$$

Phase 3: Authentication Stage 1

$$validate = g(ECPublicKeyAlice, ECPublicKeyBob, NonceAlice, NonceBob) \quad (2)$$

Phase 4: Authentication Stage2

$$Substantiate(E_a, E_b) = f3\begin{pmatrix} ECDHKey, AliceNonce, BobNonce, IOCap, \\ AliceBDAddr, BobBDAddr \end{pmatrix} \quad (3)$$

Phase 5: Link Key Calculation

$$SessionKey = f_4\begin{pmatrix} ECDHKey, AliceNonce, BobNonce, \text{“}btlk\text{”}, AliceBDAddr, \\ BobBDAddr \end{pmatrix} \quad (4)$$

2.2. Security threats in SSP

Due to the open nature of wireless media, transmission can be easily packed or cached, causing false and altered information to be inserted and delivered to piconet devices. Various official reports and researcher have accounted security problem in Bluetooth.

Jakobsson devised "Off-Line PIN crunching attack" and captured the link key by observing key establishment protocol [13]. Based on the weakness reported by Jakobsson and Wetzel [13], Aissi et al. proposed Elliptic-Curve-Diffie–Hellman (DH) key exchange pairing mechanism to build up the security of the link key [1]. To compute the encryption key Fluhrer and Lucks [9] have observed key stream and used public knowledge of the encryption mechanism in E0. Best-known attack so far against E0 devised by Luand and Vaudenay have made possible to recover the closest codeword for any linear code [16].

BT 2.1+EDR increased the strength of security by providing the protection against both passive eavesdropping attacks and Man-In-The-Middle (MITM) attacks (active eavesdropping attacks) [6]. Ever since researchers still could make the way to attack SSP. Haataja, in his Ph.D. thesis, exploited the fact that attackers gets control over the insecure channel in the first phase of pairing and modify the information to use Just Work as an association model [12]. Later it was pointed that during pairing process in passkey entry mode MITM mounts the attack and breaks the password [3,4]. Villegas [27] found Bluetooth 4.0 to be highly secure protocol and pointed that the heart of the problem resides where the two devices pair for the first time or when they have to re-establish a link key.

NIST reported about the flaws in SSP is because of Encryption key length is open to discussion, device authentication is performed but user verification is not done, Link keys are stored in non-volatile memory, influence of the pseudo-random number generators (PRNG) is not known, peer-to-peer security is not realized instead only individual links are encrypted and authenticated, Data may be decrypted in-between, security services are limited, Low-Energy pairing do not provide eavesdropping protection. It also suggested for not using Just Works association model for highly confidential data since it does not provide MITM protection [21].

2.3. Lattice-based cryptography

It has been broadly acknowledged that relying only on asymmetric cryptography based on the stability of factoring or ECDLP is surely not enough. This is mainly because of the work of Shor who revealed that factoring, DLP and ECDLP could be easily attacked with quantum computers [26]. Lattice-based cryptographic constructions clutch great assurance for post-quantum cryptography. After Shor's work on quantum factoring algorithm, various efforts were put on quantum algorithms to crack lattice problems, however, could not succeed much [26].

Ajtai started initial work on Lattice-based cryptosystems, later Lesentra worked on polynomial time algorithm for lattice problems and developed LLL algorithm and Regev worked on the Learning with Errors (LWE) problem [2,15,24]. Lyubashevsky et al. [17] used compact "ring-based" variant of LWE. Thus reduced the key size to 2–5 kilobits and proved that the cryptosystems are faster on modern hardware. Regev [24] introduced the search version of LWE and showed that it is at least as hard as quantumly approximating certain worst-case problems on lattices. Moreover, Peikert showed, search-LWE is at least as hard as classically approximating the decision shortest vector problem and variants [22,23]. Guneysu [11] presented an alternative signature scheme whose security is derived from the hardness of lattice problems. Micciancio and Peikert [18] proved the hardness of inverting the LWE function even when the error vectors have very small entries . Joppe [14] demonstrated the usage of post-quantum key exchange based on the Ring-Learning with Errors (R-LWE) by constructing cipher suites for the Transport Layer Security (TLS). Their cryptographically secure implementation, aimed at the 128-bit security level, revealed that the performance price when switching from the non-quantum-safe key exchange is not too high and provided forward secrecy against future quantum attackers.

From the literature review, it is concluded that when using lattice in cryptographic applications the size of the public key, however, is longer than EC systems. It is supposed that, given today's technologies, the increase in the magnitude of the keys is more and is compensated by the decrease in computation time.

3. Improved design of bluetooth pairing protocol: Classiquantum Resistance Secure Simple Pairing (CRSSP)

3.1. Working of CRSSP

CRSSP simulates numeric comparison mode with five phases of pairing named as Phase 1- Input-Output capability exchange, Phase 2 - R-LWE Public Key Exchange, Phase 3- Signature Generation and Verification, Phase 4 - *Second phase of R-LWE Public Key Exchange* and Phase 5 - *Link/Session Key generation.* Unlike SSP, CRSSP has single authentication stage followed by device verification. *As the Diffiee–Hellmen shared secret key computed using Lattice Based Cryptography is quantum resistance, so to strengthen the security of link key, it is wrapped with two secret keys computed in phase 2 and in phase 4 using technique R-LWE.*

At the start, Bluetooth enabled devices in piconet exchanges their IO capability and Bluetooth Device Address (BDAddr) in unencrypted form. Next Alice and Bob computes R-LWE-Diffie–Hellman (R-LWE-DH) public key and swap the public keys. After receiving the public key, both calculate the R-LWE shared secret key (Eq. (5)) from its own private key and others public key. Further, each device randomly selects a pseudo-random number of 128-bit and communicates the same to each other. Based on the secret key, nonce(random number) and public keys, pairing devices computes their digital signature (Eq. (6)), passes it to connecting device in the piconet and then substantiates digital signatures of each other. The protocol proceeds further, only when signatures are verified.

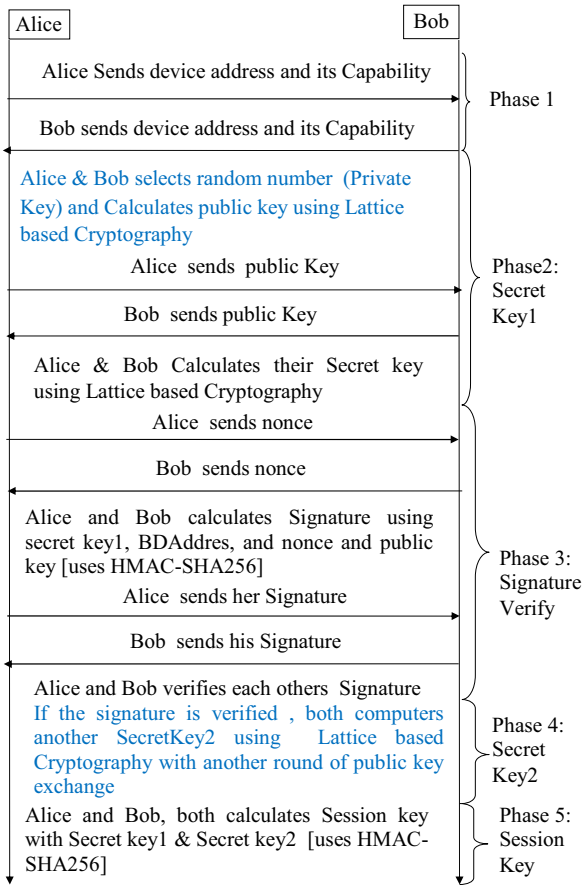

Fig. 1. Sequence diagram of proposed protocol CRSSP.

If substantiation fails, pairing begins again from Phase 1. *Once verified, both devices again compute the R-LWE shared secret key (Eq. (7)) with new public keys and masking bits. Subsequently, connecting devices calculates link/session key (Eq. (8)) from secret key computed in Phase 2 (Eq. (5)) and in Phase 4 (Eq. (7)), the publicly exchanged data in phase-4, Device address and IO capability.* To ensure that same link key is computed by pairing devices, the order of the parameters needs to be in same order This link/session key is further used to encrypt messages or file that is to be sent. Here $f_1(\ldots)$ is the Lattice-Based Diffie–Hellman algorithm, $g(\ldots)$ and $h(\ldots)$ are HMAC-SHA256 functions. All the phases of CRSSP protocol with numeric comparison mode is represented by sequence diagram in Fig. 1. It is observed that CRSSP performs total five rounds of communication during pairing.

Phase 2: Public Key Exchange 1

$$SecretKey1 = f1\begin{pmatrix} RLWEPrivateKey1, RLWEPublicKeyAlice1, \\ RLWEPublicKeyBob1, \\ RLWEMaskingBits1 \end{pmatrix} \quad (5)$$

Phase 3: Signature Verification

$$Signature = g\begin{pmatrix} RLWESecretKey1, NonceAlice, NonceBob, \\ RLWEPublicKeyAlice, \\ RLWEPublicKeyBob \end{pmatrix} \quad (6)$$

Phase 4: Second phase of Public Key Exchange

$$SecretKey2 = f1\begin{pmatrix} RLWEPrivateKey2, RLWEPublicKeyAlice2, \\ RLWEPublicKeyBob2, \\ RLWEMaskingBits2 \end{pmatrix} \quad (7)$$

Phase 5: Session Key Calculation

$$SessionKey = h\begin{pmatrix} RLWESecretKey1, RLWESecretKey2, RLWEPublicKeyAlice2, \\ RLWEPublicKeyBob2, \\ BDAddrAlice, BDAddrBob, IOCapAlice, IOCapBob \end{pmatrix} \quad (8)$$

4. Security analysis of proposed protocol CRSSP over existing SSP

The Classiquantum Resistant Secure Simple Pairing (CRSSP) protocol for numeric comparison is imitated for piconet comprising of lattice parameters m (mth cyclomatic polynomial ring), q a prime number (Upper limit for polynomial coefficients), n (Number of terms in the polynomial ring) [25]. Throughout the pairing process, devices compute lattice-based Ring Element ($n\log_2 q$ bits), lattice-based Public Key ($n\log_2 q$ bits), lattice-based Masking bits ($n\log_2 q$ bits), lattice-based Shared Secret key(n bits), Random Number (128 bits), Signature (256 bits), Verified Sign (256 bits) and Link Key (256 bits). These entities are computed even for those devices which were paired earlier. In each session, connecting devices swaps IO capability [2 bits: 00- numeric comparison mode, 10-passkey entry mode, 10-Out-of-Band mode, 11- Just work mode], Device Address, Public Key, Masking bits, Random Number and Signature for verification.

In future, high-speed computers will diminish the processing time and thus attackers will certainly crack the cryptosystems, which is still a hard problem for classical computers. Therefore, the analysis is purely inclined towards the computation and communication time of the pairing devices, to achieve a defined security level.

The CRSSP protocol is implemented with C++, and SAGEMATH library. All the computed objects is stored in SAGE file *.*sobj,* represented as file size in Table 1. Thus communication time here is the time taken to communicate the *.*sobj* file between the two devices. For both the devices, the file size in bytes, computation time of objects in seconds and messaging time in seconds with the varying set of *m, q, n* is represented in Table 1 within the context of SAGEMATH library. The data in Table 1 enlightens the fact that, for all *m, q, n* sets it is the public key and masking bits with maximum size, so is the communication time respectively.

However, data of Table 1 shows that computation time for shared secret key is maximum among all objects. In addition, the size of link key is same in all cases of *m, q, n* and is of 256 bits as CRSSP employs HMAC-SHA256 function. For all values of *m, q, n* total communication time is almost four times greater than total computation time ie communication-time = 4*computation-time. Figs. 2–4 illustrate the graph for objects' size, Computation time and Communication time of objects for one set of $m = 2048$, $q = 40{,}961$, $n = 1024$ of Table 1. On examining Fig. 2 it is found that it's the public key and masking bits with maximum size as compared to other objects but the size of lattice shared secret key is very small when compared with the public key and masking bits. Hence larger the size of public key and masking bits, harder for intruder to compute secret key, if he captures the same. Graph of Fig. 3 represents computation time of all objects and it is the secret key with highest computation time among all objects. While Fig 4 shows messaging time of each object and it can be viewed that it is the public key with maximum communication time.

Next study is focused on the existing Bluetooth pairing protocol SSP that uses ECC (Table 2) and CRSSP that uses R-LWE Lattice Based Cryptography (Table 3). Data statistics in the Tables 2 and 3 shows that to manage the security level of 256 bits, in the case of SSP pairing protocol, the size of the public key and shared secret key is same i.e 512 bits. However, in case of CRSSP to reach the same level of security, public key is 16,384 bits (Computed us-

Table 1
Communication time, computation time and size of objects of two devices for various lattice dimensions.

Lattice dimension	Object name	File size in bytes	Computation time (in s)	Communication time (in s)
m = 512	BDAddr	32	0.000203	0.0179868170000029
q = 15,361	RingElement	11,200	0.00237	–
n = 256	PublicKey1	22,800	0.007977	3.86184307199965
	Maskingbits1	20,200	0.067148	1.484711375
	SharedSecret1	2101	1.371376	–
	RandomNum	362	0.000148	0.189113595999968
	Signature	442	0.010862	0.209137411998858
	SignVerifySign	442	0.010498	–
	PublicKey2	22,700	0.007976	3.86184307187565
	Maskingbits2	20,100	0.067138	1.473711375
	SharedSecret2	2100	1.371375	–
	LinkKey	426	0.003808	–
	Total	102,905	2.920879	9.62463534387413
m = 1024	BDAddr	32	0.000335	0.0178960310000029
q = 25,601	RingElement	20,200	0.006491	–
n = 512	PublicKey1	40,400	0.016279	18.970665885
	Maskingbits1	35,500	0.169243	7.184435348
	SharedSecret1	3974	4.595783	–
	RandomNum	387	0.000166	0.188112595999969
	Signature	423	0.021107	0.0980374119988583
	SignVerifySign	423	0.021167	–
	PublicKey2	40,300	0.016179	18.923708791
	Maskingbits2	35,400	0.168243	7.184435237
	SharedSecret2	3970	4.585783	–
	LinkKey	460	0.007004	–
	Total	181,469	9.60778	45.3828560629988
m = 2048	BDAddr	32	0.000373	0.0179775050000029
q = 40,961	RingElement	38,200	0.011106	–
n = 1024	PublicKey1	76,700	0.033764	36.016444689
	Maskingbits1	65,300	0.508646	12.819412516
	SharedSecret1	7621	14.926825	–
	RandomNum	369	0.000122	0.207038269
	Signature	465	0.040863	0.260819448
	SignVerifySign	465	0.041176	–
	PublicKey2	76,600	0.032764	35.969487134
	Maskingbits2	65,200	0.506646	12.808302516
	SharedSecret2	7615	14.916825	–
	LinkKey	430	0.013756	–
	Total	338,997	31.032866	85.291179561

Table 2
Keys size for various security level in SSP with ECC [10].

Security bits	Elliptic curve cryptography	
	(Minimum size of public key and shared secret key in bits)	Link key size in bits
64	128	256
128	256	256
256	512	256

ing n*log$_2$q) and Shared key is 1024 bits (n, Numbers of Terms in Polynomial Ring) [25] which is sixteen times smaller than public key but Link Key remains same in SSP and CRSSP for all security levels. Hence in CRSSP size of the public key is log$_2$q of size of the shared secret key. Data of Tables 2 and 3 is graphically characterized through Fig 5.

Table 4 represents number of bytes communicated during pairing in SSP and in CRSSP. It is observed that, in SSP to achieve the security level of 256 bits, 3864 bytes exchanged between two devices within the context of SAGEMATH library. To attain the same level of security, in CRSSP bytes exchanged between two party within the context of SAGEMATH library is 284,666 bytes. From pure experimental results, we get bits communicated for one round of public key exchange is n log$_2q$ [7]. So for two rounds of public key exchange, bits communicated is (2n log$_2q$). Hence for two party, total bits exchanged during pairing in CRSSP is approximately (2n log$_2q$)+2[BDAddressSize (48 bits)+SignatureSize (256 bits)+NonceSize (128 bits)].

Complete result analysis is done with entire pairing time of SSP and CRSSP and is represented in Table 5. Simulation shows that to accomplish 256 bit security, pairing time in SSP is 1.85 s. And to achieve the same security level, with $n = 1024$ bit secret key, total pairing time in CRSSP is 116 s. To realize n bit secret key, for one round of key exchange, communication complexity is 2n log$_2q$+n and computation complexity is 4(estimated by the number of multiplications in the ring) [7]. Hence for two rounds of public key exchange in CRSSP communication complexity is *2(2n* log$_2q$+n*)* and computation complexity is 2×4 respectively. The proposed protocol CRSSP generates message of $n$log$_2q$ character length i.e 8$n$log$_2q$ bits for hash functions. Also hash function SHA-256 performs 64 rounds of its compression function over 512 bits (its blocks size) at a time [20]. Thus for 8$n$log$_2q$ bits, SHA-256 performs $n$log$_2q$ rounds of compression. Both party in the protocol uses SHA-256 functions for generating signatures, for verifying signatures and for computing link key. Consequently, CRSSP performs 6$n$log$_2q$ rounds of compression. As a result approximate time complexity of the protocol

Table 3
Keys size for various security level in R-LWE CRSSP.

Security bits	Lattice-based cryptography			
	Lattice dimension	Min. size of public key in bits ($n^*\log_2 q$)	Size of shared secret key in bits (Number of terms n)	Link key size in bits
64	M = 512 q = 15,361 n = 256	3584	256	256
128	m = 1024 q = 25,601 n = 512	7680	512	256
256	m = 2048 q = 40,961 n = 1024	16,384	1024	256

Table 4
Bytes communicated in SSP [10] and in CRSSP.

Security level (in bits)	SSP with ECC	CRSSP with Lattice-based cryptography	
	Total bytes exchanged	Lattice dimension	Total bytes exchanged
4	60	M = 32, q = 749, n = 16	17,460
64	1774	m = 512, q = 15,361, n = 256	86,636
128	1932	m = 1024, q = 25,601, n = 512	152,442
256	3864	m = 2048, q = 40,961, n = 1024	284,666

Table 5
Pairing time of SSP [10] and CRSSP.

Security level (in bits)	SSP with ECC	CRSSP with Lattice-based cryptography	
	Pairing time in s	Lattice dimension	Pairing time in s
4	0.030694452781	m = 32, q = 749, n = 16	5.532174551
64	0.851282435	m = 512, q = 15,361, n = 256	12.54551434
128	0.927118409	m = 1024, q = 25,601, n = 512	54.99063606
256	1.854236818	m = 2048, q = 40,961, n = 1024	116.3240456

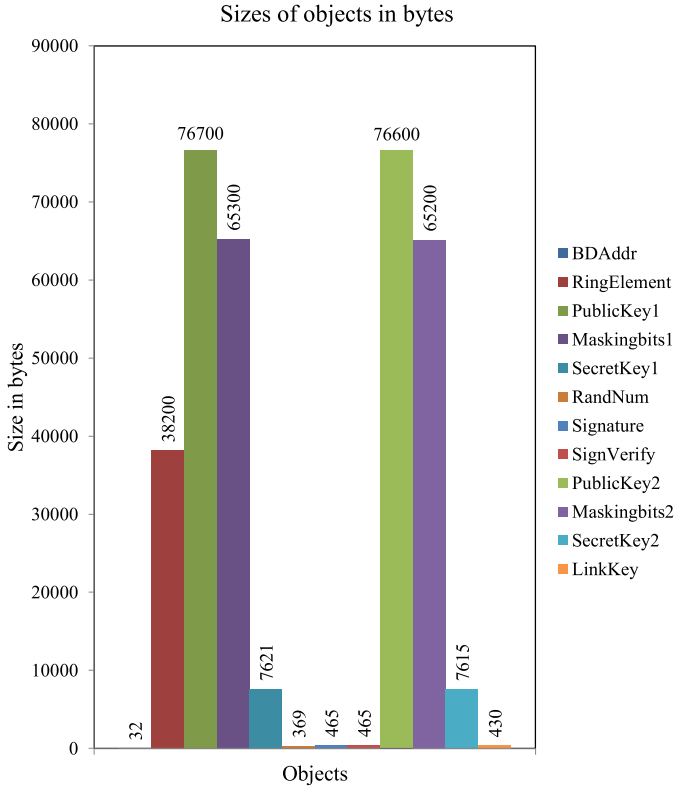

Fig. 2. Size of pairing objects for lattice dimension $m = 2048$, $q = 40{,}961$, $n = 1024$ [security level 256 bits, from Table 1].

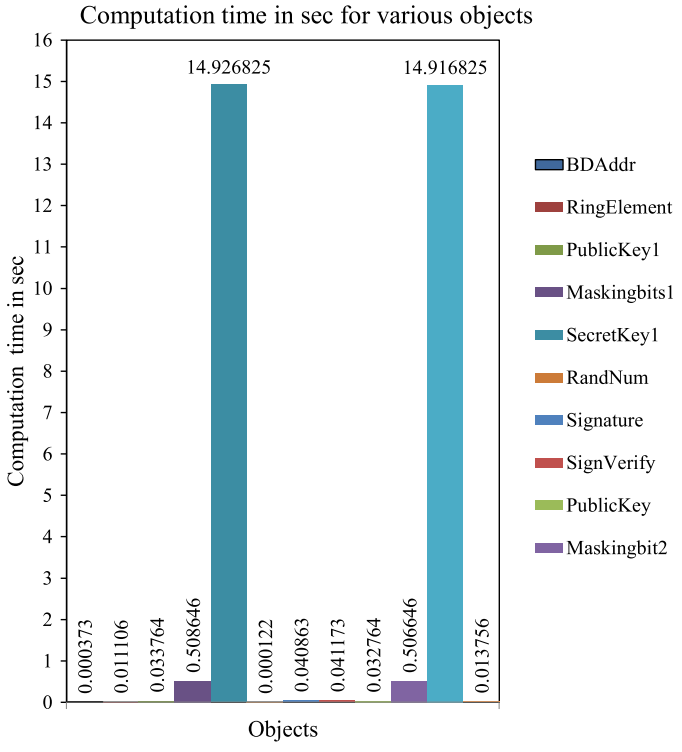

Fig. 3. Computation time of pairing objects in seconds for lattice dimension m = 2048, q = 40,961, n = 1024 [security level 256 bits, from Table 1].

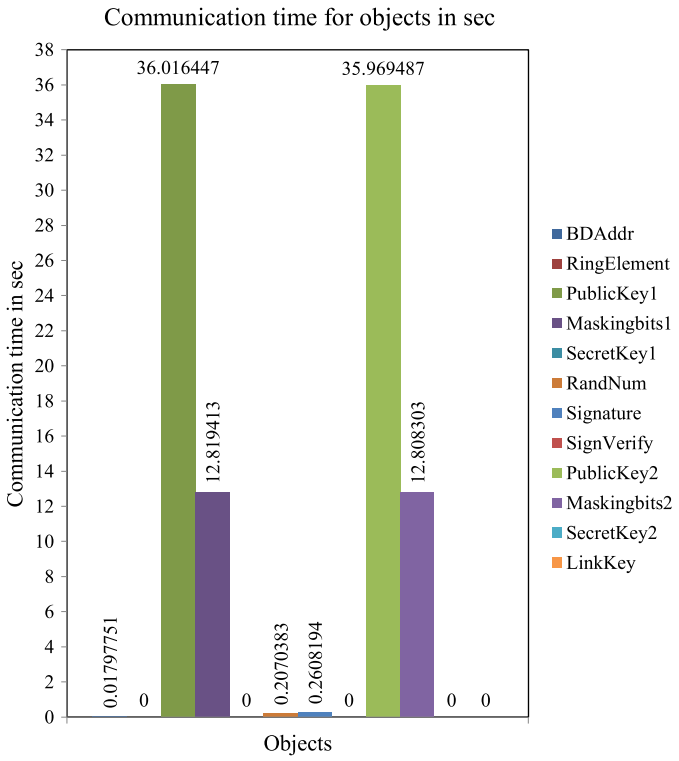

Fig. 4. Communication time of pairing objects in seconds for lattice dimension $m = 2048$, $q = 40{,}961$, $n = 1024$ [security level 256 bits from Table 1].

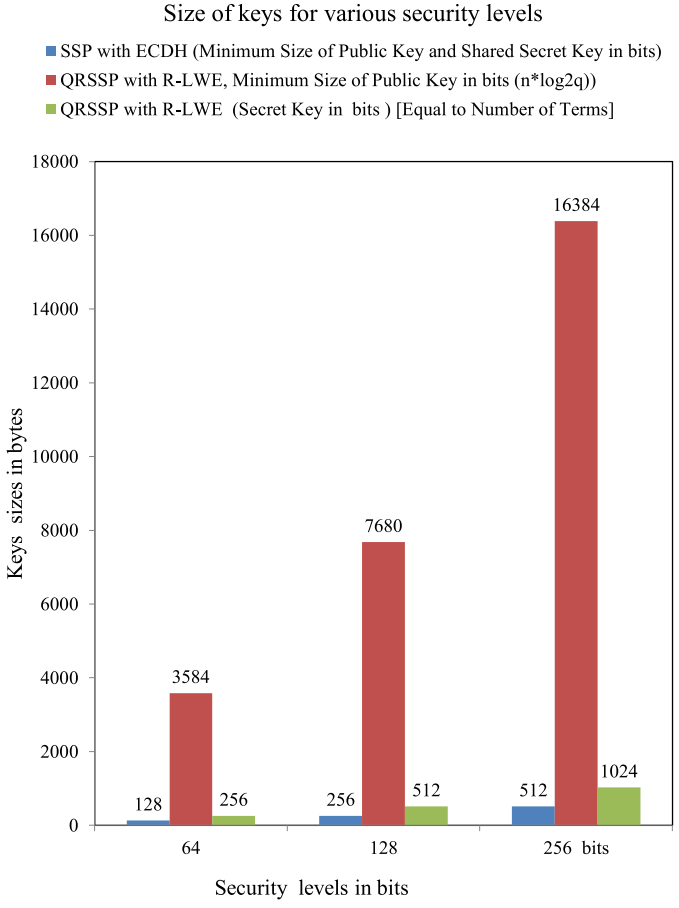

Fig. 5. Sizes of public key, secret key and link key for various security levels in SSP and QRSSP.

is $2(2n \log_2 q + n) + 2 \times 4$ (estimated by the number of multiplications in the ring) $+ 6n \log_2 q$ (rounds of compression for hash function).

5. Results and discussion

Simulation is performed on the classical computer with Ubuntu 14.04 and hardware configuration as Intel® Core™2 DUO CPU, T6570@ 2.10 GHz having 36 bits physical, 48 bits virtual address size, cache size 2048 KB and RAM 2.00GB. On wrapping the entire simulation process, it is observed that computation time, communication time, pairing time, the size of the public key in CRSSP is very large when weighed against SSP. For the classical computers, as the one which has been used, these time and size appears to be huge, nevertheless for quantum computers with increased speed, increased memory and decreased processing and computation time, it will happen in a blink of eye. In addition, if MITM somehow captures public key, for quantum computers, the inverse of R-LWE will be difficult equation for the attacker, whereas computing secret key with the inverse of ECDLP will be like a cup of tea.

CRSSP resists replay attacks in conventional and quantum computers, even for those devices, which have been paired earlier, as previously generated public key, masking bits, secret key, nonce and link key are discarded immediately after being used. Moreover, in each session, protocol computes new value for these objects. Since digital signature and its verification is realized in authentication stage of Phase 3 with challenge response mechanism, the CRSSP protocol resists passive eavesdropping attack and can discover positive MITM attack. *HMAC-SHA-256 functions and two secret keys wrap the reliability of digital signatures and link key.* It is suggested that manufactures must change the technique used in SSP to handle the pairing in quantum computers.

6. Conclusions

Security of all cryptographic systems depends on the hardness of technique that has been used to compute the key. Similarly, one of the factors for the security of SSP in Bluetooth is the hardness of ECDLP. Nevertheless, with the increasing memory and speed of computers and with decreasing computation time of CPU, its security is at risk or uncertain. Thus with this in mind to reduce the risk i.e., to increase the security of pairing between Bluetooth devices in piconet a new protocol named as CRSSP is proposed. The protocol is realized and carried out effectively for five Phases with an authentication stage and a couple of lattice-based public key exchange. CRSSP incur a small performance penalty to achieve post-quantum assurance.

It is concluded by experiments that, for classical computers, although SSP pairing time is very less than CRSSP pairing time, however for quantum computers SSP will be like a cup of tea and MITM might easily break keys in a fraction of the time. On the other hand, proposed CRSSP is a strong quantum resistant pairing protocol and it is hard problem even for quantum computers to compute the lattice based R-LWE shared secret key from the public key and hence is hard to compute the link key. Thus, we can end with the conclusion that, the CRSSP protocol shields the devices from replay attack in conventional and future high-speed computers, since each session generates fresh parameters value. Even the devices, which were earlier connected, need to be connected with fresh pairs of secret and link keys. *CRSSP averts MITM attack in classical as well in quantum computers, since link key is strongly encapsulated by two R-LWE based secret keys and HMAC-SHA256 function.*

References

[1] Aissi S, Gehrmann C, Nyberg K. Proposal for enhancing bluetooth security using an improved pairing mechanism. In: Presented to the bluetooth architecture review board at the bluetooth all-hands meeting, April; 2004. p. 19–23.

[2] Ajtai M. Generating hard instances of lattice problems. Quaderni di Matematica 2004;13:1–32 Preliminary version in STOC 1996.

[3] Andrew Y, Lindell W. Attacks on the pairing protocol of bluetooth v2.1. In: Blackhat, USA; 2008. p. 1–10.

[4] Barnickel J, Wang J, Meyer U. Implementing an attack on bluetooth 2.1+ secure simple pairing in passkey entry mode. In: 11th international conference on trust, security and privacy in computing and communications (TrustCom). IEEE; 2012. p. 17–24.

[5] SIGA BT. Specification of the bluetooth system version 2.0+EDR. Bluetooth Special Interest Group; 2001. www.bluetooth.com.

[6] SIGb BT. Specification of the bluetooth system version 2.1+EDR. Bluetooth Special Interest Group; 2007. www.bluetooth.com.

[7] Ding J, Xie X, Lin X. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology 2014. ePrint Archive, Report 2012/688 http://eprint.iacr.org/.

[8] ElGamal TA. Public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley GR, Chaum D, editors. Advances in cryptology- CRYPTO 1984, Berlin, Heidelberg, LNCS Springer; 1985. p. 196.

[9] Fluhrer S, Lucks S. Analysis of the E0 encryption system. In: Vaudenay S, Youssef AM, editors. Selected areas in cryptography-SAC 2001, Berlin, Heidelberg, LNCS Springer; 2001. p. 2259.

[10] Gajbhiye S, Karmakar S, Sharma M, Sharma S. Design, implementation and security analysis of bluetooth pairing protocol in NS2. In: ICACCI 2016, 23- 26 September 2016; LNMIIT Jaipur, IEEE; 2016. p. 1711–17.

[11] Guneysu T, Lyubashevsky V, Poppelmann T. Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff E, Schaumont P, editors. CHES 2012, Berlin, Heidelberg, LNCS Springer, 7428; 2012. p. 530–47.

[12] Haataja K. Security threats and countermeasures in bluetooth enabled systems Ph.D thesis. University of Kuopio; 2009.

[13] Jakobsson M, Wetzel S. Security weakness in bluetooth. In: Proceedings of the 2001 conference on topics in cryptology: the cryptographer's track at RSA, San Francisco, USA, LNCS Springer, 2020; 2001. p. 176–91.

[14] Joppe W, Costello C, Naehrig M, Stebila D. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: Proceedings of the 2015 IEEE symposium on security and privacy, 17-21 May, IEEE, 2015; 2015. p. 553–70.

[15] Lenstra AK, Lenstra JHW, Lovász L. Factoring polynomials with rational coefficients. Math. Ann 1982;261(4):515–34.

[16] Yi Luand, Vaudenay S. Faster correlation attack on bluetooth keystream generator E0. In: Franklin M, editor. Advances in cryptology – CRYPTO, Berlin, Heidelberg, LNCS Springer 2004; 2004. p. 3152.

[17] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: Gilbert H, editor. Advances in cryptology EUROCRYPT-2010, Berlin, Heidelberg, LNCS Springer, 6110; 2010. p. 1–23.

[18] Micciancio D, Peikert C. Hardness of SIS and LWE with small parameters. In: Canetti R, Garay JA, editors. CRYPTO 2013 Part I, Berlin, Heidelberg, LNCS Springer, 8042; 2013. p. 21–39.

[19] Miller V. Uses of elliptic curves in cryptography. In: Advances in cryptology – CRYPTO, Berlin, Heidelberg, New York, USA, LNCS Springer, 218; 1985. p. 417–26.

[20] NIST fips180-3, Federal information processing standards publication 180-3,"SECURE HASH STANDARD", October 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[21] Padgette J, Scarfone K, Chen L. Guide to bluetooth security. NIST Special Publication; 2012. 800-121.

[22] Peikert C. Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the forty-first annual ACM symposium on theory of computing, 31 May - 02 June, Bethesda, MD, USA; 2009. p. 333–42.

[23] Peikert C. An efficient and parallel Gaussian sampler for lattices. In: Rabin T, editor. CRYPTO-2010, Berlin, Heidelberg, LNCS Springer, 6223; 2010. p. 80–97.

[24] Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM 2009;56(6):1–40 Preliminary version in STOC 2005.

[25] Singh VA. Practical key exchange for the internet using lattice cryptography. Cryptology 2015. ePrint Archive Report 2015/138. See also URL http://eprint.iacr.org/2015/138 .

[26] Shor P. Algorithms for quantum computation: discrete logarithms and factoring. In: 35th annual symposium on foundations of computer science, Santa Fe, NM, IEEE; 1994. p. 124–34.

[27] Villegas J. Bluetooth low energy version 4.0. Helping create the internet of things 2012. https://www.scribd.com/document/252231581/Bluetooth-LE (Browsing Date :19/7/2014).