IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2019, accepted February 7, 2019, date of publication February 19, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900300

A Lightweight and Efficient Secure Hybrid RSA (SHRSA) Messaging Scheme With Four-Layered Authentication Stack

ANIRUDDHA BHATTACHARJYA, XIAOFENG ZHONG, (Member, IEEE), AND XING LI

Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

Corresponding author: Aniruddha Bhattacharjya (li-an15@mails.tsinghua.edu.cn)

This work was supported by the National Natural Science Foundation of China, under Grant 61631013.

ABSTRACT For virtual private network and LAN-to-LAN tunnel sessions, End-to-End security is the main constraint in private messaging scenarios. Internet Protocol Security can negotiate new keys for every communication, but when the key is compromised, it is a problem. Perfect Forward Secrecy is the resolution. In addition, the messaging scheme should be efficient and lightweight for keeping parity with daily needs. The popular Rivest-Shamir-Adleman (RSA) cipher is omnipresent in secure communications but has many scientific problems. So here, in this paper, a lightweight and efficient Secure Hybrid RSA (SHRSA) messaging scheme with four-layered authentication stack is implemented and analyzed. The scheme is resolving the problem of asymptotic very low speed of decryption of RSA, the computational modular exponentiation complexity and partial key exposure vulnerability issues of RSA, and many more. The four-layered authentication stack have eliminated the need of the use of any password, external digital certificates, and a third party for authentication with its own four techniques in a staked way. We have found that in evolutions and analysis of the scheme, it is not only resolving various scientific problems of RSA but also occupying 2%–4% less CPU than main RSA and occupying 1%–3% less memory than main RSA. Its decryption average time has gained 8.858 times compared to the main RSA and gained 2.248 times compared to CRT RSA. We have found that the RSA, CRT-RSA, and SHRSA's encryption throughput are alike–all are around 6 KB/Sec but decryption throughput of SHRSA has gained 8.5345 times than main RSA and gained 2.1174 times than CRT-RSA. The results obtained have shown us the relevancies of the SHRSA messaging scheme to be integratable as a cipher in Blockchain architectures, cyber-physical systems, and the Internet of Everything.

INDEX TERMS SHRSA, 4-Layered authentication stack, SHRSA encryption, SHRSA decryption, lightweight, encryption throughput, decryption throughput.

I. INTRODUCTION

In IEEE, 1363-2000 annex D.5.1, we have got the idea about two party forward secrecy and IEEE project P1363a has specified additional techniques. We know that Diffie-Hellman (DH) [1], [20] can offer us the PFS [20], [24]. Using Older Diffie-Hellman without curves (DHE) -PFS grade 4 [20], [24], we can be able to protect any messaging scheme from-

1. Non-repudiation attack.
2. Man-in-the middle attack (MITMA).
3. Chosen Cipher Attack (CCA).
4. Replay attack.

Now we are heading for an era of deep explorations of Blockchain architectures, CPS based production and other systems along with IoE. Nowadays we have more practically focused technique called "*Digital Twin*". We know this Digital Twin technique has a significant role in CPS- based production system. It is not an astonishing thing that most vendors of Internet of Things (IoT) platforms have implemented some form of a digital twin. The significant and relative example of this trend can be found in the Gartner's report titled "Top 10 Strategic Trends for 2017" (October 2016), Digital Twins was Number 5 strategic trend for 2017 in

The associate editor coordinating the review of this manuscript and approving it for publication was Remigiusz Wisniewski.

this report. The term "Digital Twin" was named by Dr. Michael Grieves at the University of Michigan during 2001-2002. In his paper, he brought the concept of a "Digital Twin" just same as a virtual version of what has been manufactured. Dr. Grieves defined Digital Twin Prototype (DTP), Digital Twin Instance (DTI) and Digital Twin Aggregate (DTA). The digital twin capability has three tools - Conceptualization, Comparison and Collaboration.

In recent era, there have been significant advances in the capabilities and technologies of both the data gathering of the physical product and the formation and depiction of the virtual product, the *Digital Twin*. But if we deeply look into these architectures we can find that during data communication the *security of E2E path* is very less and *very much vulnerable*. In these kinds of architectures we have to put pure *E2E encrypted data communication channel*. So we need to implement *E2E secure communication protocols* for these technologies related to Blockchain architectures, CPS, IoT and IoE. In following paragraphs we have explained the security problems, which are needed to be considered in future.

In the present Internet and in Future Internet Architectures also, we all need to protect the instant messaging information from third parties. It's a matter of the personal message's privacy and protections [1]–[10], [12]–[17], [21]–[26], [29], [38], [43], [44]. We have to protect privacy along with authentication of the users and messages; so E2E [34], [36] privacy, authentication along with strong security are the *major challenges* [34], [36]. Now for personal messaging scenario, we need *E2E encryption* [34], [36] with high efficiency. We need high authenticity in communication and if it is less CPU and less memory occupier then it is more relevant for daily use. Single layer authentication of each peer can be compromised very easily. We know that Internet Protocol Security (IPsec) can negotiate keys for every real-time communication as on time. However issue is that, if that key is compromised then whole session can be revealed. So PFS [20], [24] can resolve this.

IPSec and Datagram Transport Layer Security (DTLS) are in highlights for recent times, if we see the various security dimensions, it can be summarized in Table 1.

Also apart from the issues described in Table 1, the DTLS and IPsec are not the most enhanced resolutions, to *offer proper protection to Constrained Application Protocol (CoAP)* for many reasons. Although for CPS and IoE, it has some good features but not able to give proper protections. The reasons are-

1. IPSec and DTLS necessitate extra messages to work for the security parameters and form the security associations (SAs). But the overhead and drain out of the resources of the constrained devices will be increased much more. This is very problematic for the devices in mobility in the IoT.
2. If we think about the environs of the communication among two dissimilar networks, the ideal security resolution is depend on either IPSec or DTLS. This point towards the existence and provision of these protocols, in both the source and destination networks. But this ideal idea cannot be realistic in many circumstances, particularly when we think about the fact that, the IPSec protocol has a compatibility problem with firewall's throughout in the networks.
3. Both IPSec and DTLS count on the Internet Key Exchange (IKE) and the Extensible Authentication Protocol (EAP), for setting up the secure association and sometimes any other. So it's well understood that, this point towards those all constrained devices vendors requisite to support these additional protocols (IKE and EAP).
4. The IPSec and DTLS are aimed at securing connections among two static and remote devices. So the IPSec and DTLS attempt to offer the most possible secure connection among the two ends, devoid of the Quality of Service (QoS), the network trustworthiness or any other restrictions on the end devices considerations. But in the scenario of the constrained environment, there is a need for more dynamic and sensible actions that think about the constrained type of the end devices at the time of negotiating the security parameters.
5. The IEEE 802.15.4 specification has described the payload should be 127 bytes as whole. So if we use the DTLS as security protocol, to defend CoAP exchanges, 13 bytes (out of the 127 bytes of IEEE 802.15.4 frame) has to be assigned for DTLS record. Also 25 bytes has to be used for link layer addressing information, 10 bytes for 6LowPAN (IPv6 over Low power Wireless Personal Area Networks) addressing. Along with that, the 4 bytes of CoAP header. So as an outcome, only 75 bytes are available for application layer payload. But it is not sufficient space for communicating actual data. Subsequently, one big piece of data (bigger than 75 bytes), will use additional resources from the nodes and the network itself. The reason is that, it will be broken into several pieces and sent twice. Hence, some header compression mechanisms are good solutions,

TABLE 1. A comparison of IPsec and DTLS in various security dimensions.

Security Dimension	IPSec	DTLS
Access Control	No	No
Authentication	Yes	Partially-Server Only
Non-Repudiation	Yes/No; as per the authentication method. Public Key Infrastructure (PKI) not supported by constrained devices	Yes/No; as per the authentication method. PKI not supported by constrained devices
Confidentiality	Yes	Yes
Communication Security	Yes	Yes
Integrity	Yes	Yes
Availability	Mitigation-No full defend	Yes-stateless cookie
Privacy	No	No

at the exact cases where needed. The compressing and decompressing necessities are the reason for more constraints to the nodes and network resources.

6. In the case of DTLS, some applications might necessitate security services, to be more and more customized in relation to the application or scenarios requirements. Nevertheless, if the security were applied as per the requirements of the application or scenario, it would offer to decrease the usage of existing resources and definitely would increase the network enactment.
7. In the Internet draft of ''Datagram *Transport Layer Security in Constrained Environments*'', the authors have pointed out 7 prospective problems correlated to DTLS protocol, if employed in constrained environs. The authors also have pointed out some projected workaround, to resolve these problems. Still, much works are required to make the DTLS perfect for making it a good and prospective security resolution for IoT environs.

The *Secure CoAP (S-CoAP)* is a secure variant of CoAP. In S-CoAP, the security technique is actually an integrated part of the protocol itself. With S-CoAP, security measures will be integrated into the plain CoAP transactions. So one of the good features is that, it will have its own compromise stage that think through the limits of the constrained devices. The SCoAP prerequisites to offer security for normal connection setup, in addition to that, for the case of mobility also. So in a nutshell, the advantage is that, the security will be integral part of the CoAP protocol. It's well understood that, this security is offered by other standards, so the S-CoAP should be capable to function across numerous sites and networks.

In our daily life, we all need a messaging scheme, which will protect us from several existing scientific attacks and give us high protection and authentication to the messaging data. Also we all want faster E2E encryption and strong authentication between communicating parties.

Nowadays, we have many IM schemes. They have many backlogs like- centralized system (susceptible to single point failures), very slower decryption, authentication is by only password, prerequisite a third party, insecure default settings on IM clients. Also Denial-of-Service (DoS) attack against IM clients or the server are not fully addressed as on time, impersonation using a stolen/compromised password cannot usually be stopped in password-only systems, using *Secure Sockets Layer (SSL)* connections or digital certificates incorporated IM systems are not secured these days. So we need a messaging scheme which can replace some of these backlogs.

We all need to shield our instant IM information from third parties. It's a matter of our personal message's privacy and security. We have to protect privacy along with authentication of the users and messages; so E2E privacy, authentication along with strong security are the major challenges. In the contemporary age, the *cryptography* is well-thought-out to be a good method for providing security. The main difficulty allied to *symmetric ciphers* is the exchange of key. All the interconnecting parties necessitate a shared secret key. The exchange of key to create a secured communication in between them is needed. The security of the symmetric key algorithm can be subject to the security of the secret key. The key size also is subject to the algorithm used. The key cannot be in public online. Also if a large number of communicating clients want to communicate, then the key exchange is infeasible & very hard too. All such problems are solved by the *public key cryptography/asymmetric cryptography*. In asymmetric cipher, a shared secret can be established online in between peer deprived of any necessity for exchanging any undisclosed data. User authentication and privacy of the data also have great importance. We know that most popular asymmetric ciphers also have several disadvantages.

For providing *E2E secure communication, combining different variants of RSA* can be good choice for being a public key cryptography. RSA is omni-present these days from email signatures to SSL use. However, RSA and RSA variants also have many scientific problems. RSA algorithm, the most popular asymmetric cipher, is suffering from various attacks and backlogs in this era also.

The *weaknesses of RSA* can be –

1. Exploitation of multiplicative property and exploitation of Homomorphic property.
2. Difficulty of the integer factorization problem [2], [4], [7], [54] and computational modular exponentiation complexity problem.
3. Partial key exposure vulnerability and low modular complexity with effortlessness and speediness problem.
4. Real-time key negotiation between each peer problem and parallel protection to Sniffing attack.
5. CCA, Brute force key search, and Timing attacks.
6. Asymptotic very low speed of decryption etc.

We now have many variants of RSA, which can solve any one or two problems of RSA mentioned above, but recent attacks demands a solution cipher, which should have *solutions for at least more than one or two RSA problems* [1]–[10], [12]–[17], [21]–[26], [29], [38], [43], [44]. Also that proposed scheme should replace some of the pitfalls of Instant Messaging (IM) schemes.

Therefore, we have implemented here a lightweight and four layered authenticated SHRSA messaging scheme and analyzed it in various aspects. It has four-layered authentication stack before the SHRSA encryption starts. These 4-layers of authentication stack is to ensure multiple authentications layer-by-layer resulting stronger authentication than any single authentication way. *RSA, CRT-RSA, Multi-factor RSA* and *Multi-Prime RSA*, all have single layer key exchange authentication, which is *easily breakable* nowadays.

The authentication stack of the SHRSA scheme consist of – Diffie-Hellman Key exchange- Layer 1, 3-way Handshaking- Layer 2, Peer to Peer Authentication by Diffie-Hellman Key exchange - Layer 3, and PFS grade 4 - Older Diffie-Hellman without curves (DHE)- Layer 4.

Then SHRSA encryption works, followed by SHRSA decryption. The SHRSA messaging scheme is solving many issues of variants of RSA. We know that PFS grade 4 [20], [24] has one distinctive property. An agreed key will not be negotiated. The SHRSA messaging scheme offers us strong *security, privacy* and *efficiency for E2E encrypted messaging*. *Less CPU* and *memory occupier* features of the scheme make the scheme more accepted in low CPU power and memory occupancy environs like CPS-based productions and other systems and IoE environs. Our details analyses have defended our claims in real-time testing with practical results.

The body of the rest of the paper is as follows- section II has given a brief review of present scenario correlated to the implemented messaging scheme. Section III has presented the scheme. Section IV has discussed the security analysis of SHRSA messaging scheme's cipher. Section V has discussed performance evolution and analysis of the scheme. Section VI has discussed the major contributions of the work and shortcoming of the work. Last but not the least; section VII has concluded the paper.

II. PRESENT SCENARIOS

In present scenario of secure group communication, we have group key management system, which is categorized into *three* classes. A trusted entity selects a secret key for communicating among the joining groups; this is the *first key management scheme*. When the work of trusted entity is distributed amongst subgroup managers, this is the *second key management scheme*. In a scenario, when the group members are trusted alike and all are joining in key establishment, this is *third key management scheme*. A group key management scheme has to fulfill Key Independence, Perfect Forward Secrecy and Backward and Forward Secrecy to attain security during key exchange. Depending on computational hard conventions for instance, Discrete Logarithm Problem (DLP) and Computational Diffie-Hellman Problem (CDHP), Perfect Forward Secrecy based scheme can confirm *non-repudiation*. Here, any invader cannot disclose the short-term group key, although the long term keys are accidently leaked or compromised by mistake. So *PFS based this kind of scheme* has already proven secured against some important and dangerous attacks. For instance, *non-repudiation attack, MITMA, CCA and replay attack* [20], [24].

For accomplishing User Authentication and the Key Establishment (UAKE), a common tactics is to use the public-key cryptosystem, like RSA and ElGamal. However, most relevant disadvantage is that, the public-key cryptosystem computes *modular exponentiation*, which is time-consuming operation. Therefore, it's well understood that, it is not the best choice for the scenario of low computational environs and small storage environs.

SSL has some disadvantages, which are very irrelevant in modern times like-

- It is very slow, due to handshaking for establishing connections and it involves encryption and decryption of the data mutually.
- It always has overhead for deployment at server-side (as we know that protocol is resource intensive and slow).
- It sometime allows insecure encryption, as SSL enables our computers on either side of a connection to work together for choosing the cipher system they want to use. It also let an old software or a misconfigured server to choose an encryption method that is very insecure. So sometime it can be insure also.
- Getting SSL can be expensive and need to be renewed periodically.

Now let's discuss some of the secure protocols available now. Now if we see as per transmission overhead, the overhead due to connection establishment, and the processing overhead, then for IPSec, ISAKMP (Internet Security Association and Key Management Protocol) payload header = 20 + 8 + 4 = 32 Bytes (B), total (IPSec/TCP) header = 85+20 = 105 bytes (total IPSec header = 20+24+41 = 85 bytes, TCP header = source port + destination port + sequence number + acknowledgment number + control + data offset + reserved + window + checksum + urgent pointer = 16+16+32+32+6+4+6+16+16+16 = 160 bits = 20 bytes (TCP with no options)), and total connection establishment delay = IPSec RTTs + TCP RTTs (Round-Trip Time) = 4.5 + 1.5 = 6 RTTs. For case of *Transport Layer Security (TLS)*, TLS handshake packets have total of 32-bit header, every single handshake record has a 9-byte (with record size) overhead plus a TCP header, which is 20 bytes. Total TLS header = 40 bytes and total Header of TLS/TCP message = 60 bytes [TCP header = source port + destination port + sequence number + acknowledgment number + control + data offset + reserved + window + checksum + urgent pointer = 16+16+32+32+6+4+6+16+16+16 = 160 bits = 20 bytes (TCP with no options)]. TLS/TCP handshake necessitates 1.5 RTTs (TCP) + 2 RTTs (TLS) = 3.5 RTTs for forming a session. An assessment of the header of a handshake message (in bytes) in TLS and DTLS can be as shown in Table 2. Point to be noted is every handshake record has 25 bytes (with record size) overhead in addition to the Stream Control Transmission Protocol (SCTP) header.

TABLE 2. TLS vs. DTLS handshake comparison.

Field	TLS	DTLS
Message type	1	1
Message length	3	3
Message sequence number	Doesn't exist	2
Fragment offset	Doesn't exist	3
Fragment length	Doesn't exist	3
Total	4	12

Also for DTLS, SCTP header = source port + destination port + verification tag + checksum = 16 + 16 + 32 + 32 = 96 bits = 12 bytes, total DTLS header = 48 bytes and total header of a DTLS/SCTP message = 60 bytes. DTLS/SCTP handshake necessitates 2 RTTs (SCTP) + 3 RTTs (DTLS) = 5 RTTs for forming a session connection.

It's well known that minimum size of a Diameter message with a single Attribute-Value Pair (AVP) is 32 bytes.

So concisely, we can find following things-

- ✓ IPSec encrypts the IP (20 bytes), TCP (20 bytes), Diameter headers (32 bytes), and ESP trailer (2 bytes assuming no padding), a total of 74 bytes.
- ✓ TLS encrypts the hashed data MAC (20 bytes), TCP (20 bytes), Diameter (32 bytes), a total of 72 bytes.
- ✓ DTLS encrypts the MAC (20 bytes), SCTP (12 bytes), Diameter (32 bytes), for a total of 64 bytes.
- ✓ In IPSec hashing is done over ESP with the extra ESP header (8 bytes): 82 bytes.
- ✓ TLS hashing is done on TCP (20 bytes) and Diameter (32 bytes): 52 bytes
- ✓ DTLS hashing is done on SCTP (12 bytes) and Diameter (32 bytes): 44 bytes.

Here one point to mention is that, depending on the *encryption* or *hashing algorithms selection* by the sender and receiver, we can have dissimilarities in the delay and header. Fact is that, the processing is highly reliant on the *hardware* used, mainly in the environs of hardware accelerators. We have seen the processing overheads in all cases like *IPSec/TCP, TLS/TCP, DTLS/SCTP, TCP, SCTP* are almost of the alike order. Only *IPSec* is with the exception, its hashes almost double the amount of bytes. So IPSec and SSL all have their limitations, so we need *alternatives to replaces these limitations*.

We have seen that, IPsec 6to4 transition mechanism displayed nearly three times delay compared to its original value. 6to4 with IPSec has shown the highest overall TCP DNS throughput. Another aspect was, for 4to6 UDP (User Datagram Protocol) DNS (Domain Name System) throughput is lesser than 4to6 with Point-to-Point Tunneling Protocol (PPTP) VPN, but in case of packet size 512, we have seen slightly higher throughput. 6to4 with IPsec has shown the lowest UDP DNS throughput. So it's clear that with IPSec, we have different results which are not good.

Now matter is, if we go with Cryptographic approach, they have many problems. The security of the symmetric key algorithm is subject to *the security of the secret key.* Also in cryptography if a large *number of communicating parties* want to communicate, then the key exchange is infeasible and very hard too. Now if we consider the most popular Public Key cryptography RSA, it also has many problems. RSA's encryption and decryption operations are very costly. The elementary E2E [34], [36] security services, like instance freshness of secret keys among two communicating entities, authentication, privacy and confidentiality are obligatory. CoAP/CoAP, DTLS/DTLS and HTTP/CoAP, TLS/DTLS, these kinds of different usage scenario make the trouble to achieve the E2E encryption. Also these cryptographic approaches *are not secure from various attacks* as we have found in our past works [45]–[53] like, CCA, Brute force key search, Timing attacks and Mathematical attacks etc.

IM schemes nowadays have many backlogs. Some of the IM schemes' backlogs are-

1. Centralized system, so single point failure can occur anytime, anyplace.
2. Only messages are encrypted, not strong encryption for communicating party's communication protocol.
3. Decryption is not faster. Statistical complexity is less and vulnerable to CCA and other attacks.
4. Authentication is by only password.
5. The third party use is a disadvantage. Nowadays lots of IM schemes have third party-based security.
6. Insecure default settings on IM schemes for clients are a big problem.
7. Sharing IMs features with other applications introduce significant security risks.
8. DoS attack is a big problem.
9. Pure Peer to Peer (P2P) scheme is used in very less cases.
10. The SSL-based solutions for public IM service have drawbacks.

Let's now discuss some of the issues about attacks in CPS and IoE-.

A. SECURE SERVICE MANAGER (SSM) SPOOFING ATTACK

If an attacker is the SSM, then most dangerous thing is that, the attacker can acquire all the information about the session, due to the reason of delegating the DTLS handshake. So there is a chance that, the encrypted data among end nodes can be exposed to the attacker. A good solution can be, use of PSK_DN (which is shared among the SSM and a constrained device in the bootstrapping phase). The good reason for this protection of the SSM Spoofing Attack is, data is encrypted by use of PSK_DN and then sent, and the attacker cannot deceive a constrained device and cannot get the right to use the encrypted data.

B. SEMI E2E SECURITY

We have to ensure E2E security. The SSM can acquire all session information, by just delegating the DTLS handshake. As we know the encrypted session information are sent to a constrained device instantly but, the SSM does not do the accumulation of session information. So it's well understood that, end nodes joining in the DTLS communication will encrypt and decrypt data themselves only. The SSM is only responsible for the data relay after sending the session information to the constrained device. In this kind of system, the executor of the encryption and decryption is the end node in the DTLS communication. There is one obligatory thing; the SSM must trust the pre-registered device for example smart phone of user. So as an outcome, we get an E2E security (semi E2E security exactly) definitely. But it's a type of semi-E2E security.

C. DoS ATTACK

The devices setting up IoT, have low CPU performance and a small amount of memory. So it's a well understood fact that, sending a DTLS handshake request message to these low memory and low performance devices can seem to be a *DoS attack*, even supposing the request is from a legitimate user. Another case is, if an attacker transmits a DTLS handshake

message straight to a constrained device with conditions in the low-power lossy networks (LLNs), then as an outcome, the devices become more dangerous. So we can understand that, the SSM benefits to resolve the DoS issue by delegating the DTLS handshake. The SSM stops constrained devices from receiving a lot of messages directly.

D. SINGLE POINT OF FAILURE

Numerous methodologies applying delegation, can give a *single point of failure (SPOF)*. It is one of the utmost predictable, but serious difficulties in security field. We know that, the SSM has a significant role of delegating DTLS handshake in place of numerous CoAP sensors. So it is well understood that, if the SSM is negotiated or fails then, all the sensors under the SSM cannot create a secure session with client or server, which are outer of the LLN.

A well-defined trust manager can somehow protect such a SPOF issue. The trust manager has the option to choose alternative authentic device, as a new SSM. Then he can broadcast associated information to his sensors. Only thing is that, the SSM should be resource rich devices in smart home or smart building (e.g. smart healthcare devices etc.). Another way can be, virtually applied SSM in cloud system. It is harder to compromise a virtual SSM in Cloud, as it is operated and supervised by security manager, compared to attack a home device or smart phone, which is operated by its usual user. One highlighting point is that here, a secure registration method between the SSM and IoT devices controlled by the SSM is there. Moreover, another supposition is that, the secret key, which is common for both SSM and its devices, cannot be compromised. Future research can be designing and implementing a concrete secure system, with additional mechanisms including key revocation, secure bootstrapping, trust management, and so on.

E. FRAGMENTATION ATTACKS

A packet fragmentation mechanism is a good resolution for dissimilar MTU (maximum transmission unit) size among Internet and LLN. An IPv6 adaptation layer 6LowPAN, has a provision with a method to fragment large IPv6 packets into small frame. Normally, sensing data and control data for actuators can be small in size. Though, DTLS handshake message is bigger in size than the maximum frame of LLN in size, for instance IEEE 802.15.4 (i.e. 127 byte). Particularly, DTLS fragmentation is unavoidable at the 4th flight of DTLS handshake. The reason is that, it encompasses comparatively large size of certificate of server and key exchange message. We can send 27 DTLS fragmented datagrams in case of using TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 with Raw Public Key Certificate. Significant transmission overhead is the outcome from these fragmented datagrams for the reason of the header added to each of the frames. But some other critical issues are that, due to the deficiency in authentication mechanism at 6LowPAN layer, it gives chance for attackers to try buffer reservation attack, fragment duplication attack and fragmentation attacks. An attacker eavesdrops and modifies a fragmented frame in the middle of the wireless multi-hop link to lunch the fragment duplication attack. At the time of receiving, the Target node cannot identify the altered frame. So as an outcome, the attacker's just a single forged frame can stop successful reassemble execution of the target node. Additionally, the target node requisites to abandon all frames in the buffer and waits for retransmission once more, resulting the DoS attack. We know that, the first frame retains a memory space for reassembling the original packet and it is indicated in the header (i.e. datagram size field) at the target node. Also the buffer reservation attack exploits this fact. The attack can be very simple, like the attack can be done by sending a forged start frame encompassing large number in the datagram size field. A good option with a good efficiency can be a scheme, which uses the SSM to delegate the DTLS handshake phase. For the constrained network like a LLNs, network overhead, and delay and loss problems, due to fragmented handshake message packets, is resolved by delegating the handshake. For the constrained device, the device need not to retain the fragmented handshake packets, in the buffer up to the receiving all of them. In addition, DTLS communication devoid of any source code for a DTLS handshake can be used by a constrained device. Here the E2E security is definite, as data encryption and decryption are done in the end node. Also it's more important feature is that the system can tackle an SSM spoofing attack and DoS attacks on a constrained device. Another highlight is that, the SSM and the constrained device are tangibly distinct, but can virtually be considered one system in a trusted relation with a shared key. This shared key is a pre-shared. So in a nutshell, this kind of scheme can benefit to deploy constrained devices in a secure manner in constrained environments.

Now the issue is *balancing strong security, privacy, reliability, authentication, efficiency* and *less CPU* and *memory occupier property*, which is tough. Also as per the authenticity demand nowadays, we need *multi layered authentication*; single layer authentication cannot be so strong like Diffie-Hellman only cannot be strong enough for authentication. *Efficiency* is also a pivot element for any kind of messaging scheme. RSA suffers some very dangerous problems like, exploitation of multiplicative property [3], homomorphic property [3], [35] (*MITMA*), difficulty of the integer factorization problem, very computationally costly exponentiation modulo N, computational modular exponentiation complexity, partial key exposure vulnerability, asymptotic very low speed of decryption, Sniffing attack, real-time key negotiation between each peer problem and effortlessness and speediness problem. Variants of RSA works in the past as discussed in [45]–[54], can play an important role to resolve these issues for the existing Internet and Future Internet Architectures [28], [37].

Therefore, it is very clear that, we need to address these most important issues and we need a secure messaging scheme, which is *less CPU* and *memory occupier, efficient* and *with a strong privacy feature with very*

high-level authentication. It should also replace some of the disadvantages of existing IMs as discussed.

III. A LIGHTWEIGHT AND EFFICIENT SECURE HYBRID RSA (SHRSA) MESSAGING SCHEME

The SHRSA messaging scheme is an outcome of Main RSA, Shared RSA, Efficient RSA, CRT RSA and Multi-Prime RSA. For VPN and LAN-to-LAN scenario, we need pure E2E encryption. The scheme is an installable software form now [46], [48]. Both in the peer (users) have to install the messaging software at first. After installation complete, user can start the messaging scheme just by clicking on the applications. Our four layered authentications work in stacked way as shown in Figure 1. The four layered stacked type authentication works before SHRSA personal messaging starts. These multi layered authentication stack eliminates the chances of *any keys negotiated online for peer*, who are connected for personal messaging. So we have these extra 4 layers of authentication apart from main SHRSA's authentication, to strongly ensuring the authentication of each peer. We know that above variants of RSA which are seed for SHRSA messaging scheme, all have their own single layer authentication.

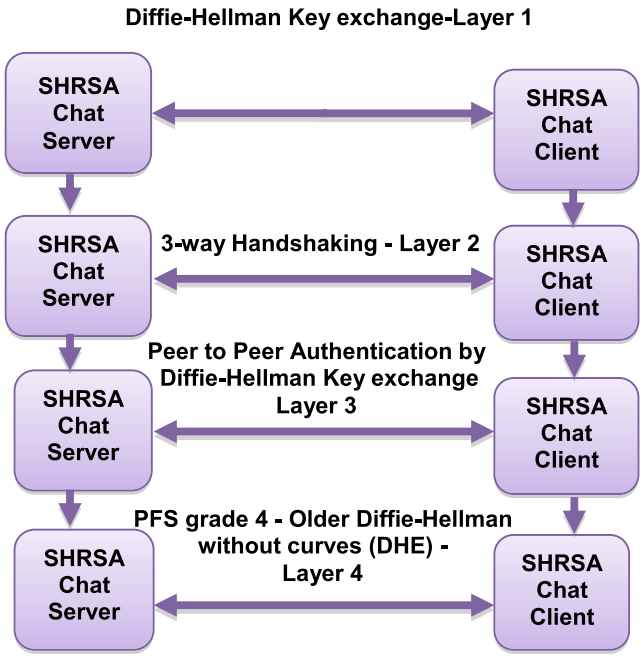


FIGURE 1. Four-layered authentication stack for SHRSA messaging scheme.

Here in the Figure 1, in layer 1, we are using simple Diffie-Hellman Key exchange for both SHRSA peer (considered in the scheme as SHRSA server and SHRSA client). We are verifying the peer by our own designed challenger class with unique IP address by use of *Trusted Third party* technique [TTP]. We can have n number of SHRSA clients but the SHRSA scheme's architecture is designed in such a way that, all other SHRSA clients, who wants to message, have to wait for P2P authentication. Therefore, they can connect to the SHRSA server, but could not initiate the messaging. This way the architecture ensures pure P2P authenticated messaging, before the main SHRSA messaging scheme starts.

In layer 2 of the four-layered authentication, we are using the 3 way handshaking between the peer. Then we are using main Diffie-Hellman Key exchange protocol for layer 3 in 4-Layered authentication stack. After that in layer 4, we are using PFS grade 4 - Older Diffie-Hellman without curves (DHE) [1], [20], [24]. Here the reason for using the PFS is its unique property's advantage. Its unique property is an agreed key will not be negotiated, even though agreed keys are nothing but the derivative from the same long-term keying material, in environs where, subsequent run are negotiated. So this way the 4 layered authentication stack is resolving the disadvantages of IPsec. The DH - PFS based 4-layered authentication stack defends the keys in transit, averting identity release to passive Sniffing attack at network level.

Also we have four times real-time key negotiations in this 4 layered stacked architecture, which is *eliminating the change of identity release*. By functioning at the network layer, IPsec can be used with any transport layer protocol comprising TCP, UDP, HTTP, and CoAP. IPsec confirms the confidentiality and integrity of the IP payload by use of the Encapsulated Security Payload (ESP) protocol and integrity of the IP header plus payload by use of the Authentication Header (AH) protocol. IPsec is obligatory in the IPv6 protocol, so all IPv6 equipped devices by default have IPsec provision, which may be facilitated at all time. Being obligatory in IPv6, IPsec is one of the utmost appropriate alternatives for E2E security [34], [36] in the IoT [29]–[31], as commonly only one application runs on a constrained device and the generic security strategies are adequate for such circumstances. Additionally, application developers necessitate reasonably slight effort to facilitate IPsec on IPv6 hosts, as it is already applied at the network layer by device vendors. But as IPsec has a huge number of features with lots of options, so it is too much complex. This complexity raises the probability of the occurrence of a *weakness*; IPsec is weak *against replay attacks*. In this replay attacks, valid data transmission is *illegally repeated or delayed.* Also IPsec is more complicated to put into practice to individual users on multi-user machine, so for instant messaging scenario *it's not at all a good choice*. So in a nutshell, the *4-layered authentication stack* in the SHRSA messaging scheme is able to defend the *keys in transit, averting identity release to passive Sniffing attack at network level*. The four times real-time key negotiations in this 4 layers architecture are also *able to eliminate the change of identity release*. In the messaging scheme where ever prime numbers are needed, in all places, we are using *Rabin-Miller primality test (GARY L. MILLER)*, due to the properties of strong pseudoprimes, it's very unique.

So this *four layered authentication stack* in the scheme is resolving many problems which main RSA, CRT-RSA, Multi-prime RSA and Multi-factor RSA do not solve. Main RSA, CRT-RSA, Multi-prime RSA and Multi-factor RSA are

only with *one layer key-exchange,*that authentication can be broken anytime.

The SHRSA's 4 layered authentication stack are providing protections from the following attacks- *non-repudiation attack, MITMA, CCA, replay attack, giving us 3-way handshaking, both in the peer are authenticated by key exchange in every layer, parallel protection to Sniffing attack and real-time key negotiation between each peer*.

WE are using *Square and Multiply Algorithm*, in SHRSA encryption, for *decreasing the high cost of modular exponentiation computation*, of a^c mod n for Main RSA as stated in Algorithm 1.

Algorithm 1 Square and Multiply Algorithm

Input: $a, n, c = c_{k-1}c_{k-2}\cdots c_1c_0$.
Output: Square-and-multiply (a, n, $c = c_{k-1}\ c_{k-2}\cdots\ c_1\ c_0$)
$z = 1$
for $i \leftarrow k\text{-}1$ down to 0 {
$z \leftarrow z^2 mod\ n$
if $c_i = 1$ then $z \leftarrow (z \times a)\ mod\ n$
}
return z

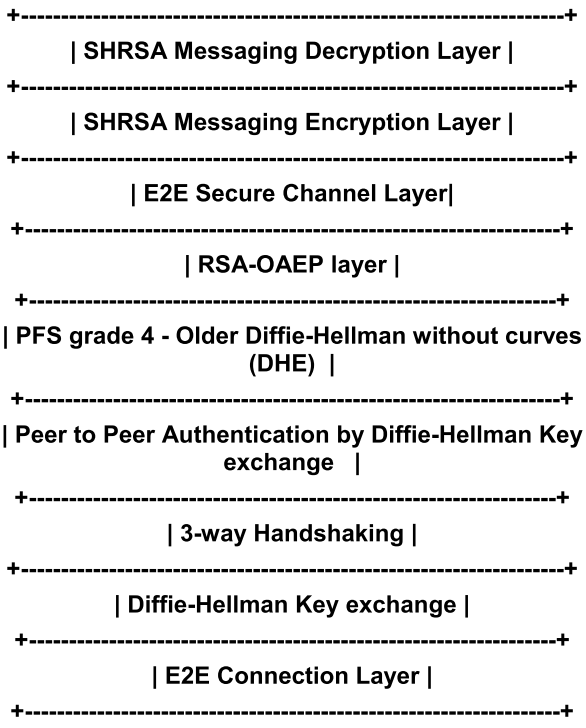


FIGURE 2. SHRSA nine layered protocol stack.

The efficient and four layered authenticated SHRSA messaging scheme's protocol stack is actually a *nine-layered stack*, as shown in Figure 2. in the protocol stack as shown in figure 2, *E2E connection layer* is responsible for creating and receiving TCP connections from peer with TCP relay option, recognized by ip address used. for more clarity on 4 layered authentications stack we have highlighted that in details in Figure 1. so Figure 2 is the whole scheme's 9 layered protocol stack *which works over TCP* and it also has included the 4 layered authentication stack (layer 2- to layer 5).

Now in the *sixth layer* we are using Optimal Asymmetric Encryption Padding (OAEP) [32] and random salts with the plain text. This OAEP and random salts are helping us resolving many major scientific problem of RSA. It is protecting us from the scientific attacks like - *exploitation of multiplicative property (CCA)* and *exploitation of homomorphic property (MITMA)* and *Short Plaintext Attack.* Again *factorization of RSA modulus n* and *Chosen Plaintext Attack* are prevented by the salting process with OAEP in the messaging scheme's 9 layered protocol stacks. So in a nutshell the sixth layer - OAEP with random salts added on runtime with synchronize time gap in the SHRSA messaging scheme's 9 layered protocol stack, are resolving following scientific problems- *exploitation of multiplicative property (CCA), exploitation of homomorphic property (MITMA), Short Plaintext Attack, factorization of RSA modulus n and Chosen Plaintext Attack.*

Also the sixth layer with OEAP [32] insertion with random salts is helping us to stop following dangerous attacks - *Algebraic Attacks, Hastad Attack, Desmedt-Odlyzko Attack, Related Message Attacks, Fixed Pattern RSA Signature Forgery and Two Attacks by Bleichenbacher [32].*

After this RSA-OAEP operation, E2E Secure Channel Layer is the seventh layer; this layer makes the virtual channel streams, which is accountable for E2E encrypted tunnel from one peer to one more peer for carrying secure, reliable and strong private messages. Then the SHRSA messaging scheme's *encryption* of messaging starts. As the estimated complexity necessitate for the 1024 Bit RSA modulus is approximately 2^{80} operations, in daily life messaging scenario it's infeasible to break. So it can be considered now very much infeasible in our daily life, so we are also using 1024 Bit RSA modulo.

SHRSA messaging scheme's Encryption Layer is in charge encryption of real-time messages in every peer to peer chat session. Actual messaging starts with SHRSA encryption. In the encryption, we have first integrated main RSA with Pohlig-Hellman Encipher and with Efficient RSA [27], [41], for more and more strong and statistical complexity. The scheme is having more and more effortlessness for users with high speed by use of customized Efficient RSA with Extended Euclidean Algorithm. Here Efficient RSA with Euler Phi function that is Euler's Totient is used. The totient of n, $\phi(n) = (p-1)\cdot(q-1)\cdot(r-1)$, where $n = p*q*r$, where p, q and r are three primes. So it's customized Efficient RSA. Beastliness of this cipher is that the totient is the count of the number of elements that have their gcd with the modulus equal to 1. This expresses us to a key equation regarding the totient and prime numbers: $p \in P, \phi(p) = p - 1$.

We have used the definition for the Euler Phi function as below in the Equation 1-

$$\begin{aligned}\phi(n, h) = {} & (p^h - p^0)(p^h - p^1)\cdot\cdot(p^h - p^{h-1}) \\ & + (q^h - q^0)(q^h - q^1)\cdot\cdot(q^h - q^{h-1}) \\ & + (r^h - r^0)(r^h - r^1)\cdot\cdot(r^h - r^{h-1}) \qquad (1)\end{aligned}$$

here h is randomly picked among the integers mod n, ever since this function is in use only in the key generation process and the encryption and decryption processes are identical to the original RSA, the computational rate due to encryption and decryption will not be different ominously from the original RSA.

The scheme's encryption works very effortlessly and speedily with the use of *Extended Euclidean Algorithm in the efficient RSA like above in Equation 1*. So for resolving the *effortlessness and speediness problem of RSA encryption*, we are using *customized Efficient RSA with Euler Phi function* with the *Extended Euclidean Algorithm* in SHRSA scheme's encryption without any extra cost, here also the SHRSA messaging scheme's encryption complexity is $(3n_e - 2)(n^2 + 2)$.

The *SHRSA encryption* followed by decryption works as below-

The RSA modulus was modified so that it can further decrease the decryption time. It consists of k primes $p_1, p_2, \ldots, p_k$ instead of using only two.

For compute we have consider k=3,

Following parameters are used:

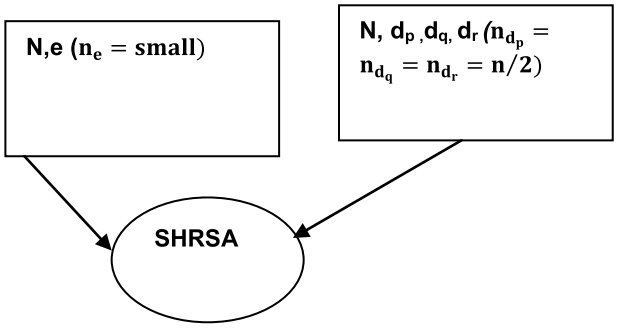
- n = No of bits in modulus.
- n_e = No of bits in public exponent (e).
- n_d = No of bits in private exponent (d).

Key Generation Method: k = No. of primes to be used.

1. Compute k distinct primes $p_1, \ldots, p_k$ each one $[log\ N/k]$ bits in length and $N = \pi_{i-1}^{k} p_i$.
2. Compute e and d such that $d = e^{-1}\ mod \varphi(N)$, where $gcd\ (e, \varphi(N)) = 1$, $\varphi(N) = \pi_{i-1}^{k}(p_i - 1)$
3. For $1 \leq i \leq k$, compute $d_i = d\ mod(p_i - 1)$.

Public key = (N, e).

Private key = $(d_1, d_2, \cdots, d_k)$.



The *advantages of SHRSA messaging scheme's E2E encryption* are as follows:

1. SHRSA encryption is based on true E2E encryption, resulting the text messages getting protection for confidentiality, integrity and availability everywhere all the time. So it maintains CIA (Confidentiality, integrity and availability) triad.
2. The same key is used for encryption and decryption, no need of a *separate key* for the decryption of the packet data (text) in the hop computer on the network.
3. Much secure (E2E) in case of sensitive data.
4. Higher modularization of the functionality is quite possible in SHRSA messaging scheme.
5. The file size involved in SHRSA messaging scheme is smaller less than 0.250 kb, and the messaging processing uses very minimal yet sufficient set of resources and encryption time, just alike main RSA with additional effortlessness.

These ways the scheme's encryption gives us following *solutions*-

1. Enable us to tackle dangerous *known-plaintext attacks*.
2. It is giving us more and more strong and statistical complexity.
3. Parallel protection to Sniffing attack and real-time key negotiation between each peer are given by PFS grade 4 - Older Diffie-Hellman without curves (DHE).
4. It is having more and more effortlessness for users with high speed by use of Efficient RSA with Extended Euclidean Algorithm.
5. The computational rate due to encryption and decryption will not be different ominously from the original RSA. So the SHRSA encryption is also having the Encryption Complexity $(3n_e - 2)\ (n^2 + 2)$ alike main RSA.
6. It enables us to be protected from exploitation of multiplicative property (CCA) and homomorphic property (MITMA).
7. It provides us the resolution for difficulty of the integer factorization problem of RSA and very computationally costly exponentiation modulo N problem.
8. 1024-bit key is giving us solution for the exploitation of certain key choices problems.
9. RSA's high computational cost is totally irrelevant in all aspects of the IoE scenarios [28], [37], [39]. This high cost owing to modular exponentiation is getting decreased by use of the Square and Multiply algorithm in the SHRSA scheme's 9 layered protocol stack.

Now let's focus on the 9th layer of the messaging scheme's 9 layered protocol stack. SHRSA Decryption Layer is responsible for the SHRSA decryption of real-time messages in every chat session.

In the SHRSA messaging scheme's decryption, at the beginning, we have integrated the main RSA scheme with the Shared RSA [41], which has made the *Hybrid decryption more complex and statistically problematic*. As we know RSA has a big problem called as *Low Modular complexity*. So by using the Shared RSA, the SHRSA Decryption not only giving solution to Low Modular complexity problem but also has made the SHRSA Decryption stronger, more complex, more challenging and difficult to be broken.

Then ultimately to enhance the decryption speed we have used CRT-Multi-Prime RSA [33], [41], [42]. The SHRSA decryption has attained a significant decryption speedup compared with plain RSA and CRT RSA [11], [19], [20], [44], just by decreasing the *size of exponents and moduli,* at the *cost of extra modular exponentiations*. Though, a linear rise in the number of exponentiations

turns to a *cubic reduction* in the cost of each exponentiation, for a complete speedup that is *quadratic in the number of factors k of the modulus*. Properly, evaluating C^d mod n for $d = O(n)$ costs $O(\log^3 n)$, while SHRSA has $d_i = O(n^{1/k})$ (so that $\log d_i = O(\log(n)/k)$) and multiplication cost of $O((\log(n)/k)^2)$ for an complete cost of $O(k(\log(n)/k)^3) = O(\log^3(n)/k^2)$. We have used 3 prime numbers (k=3) for better security then make them go through divide and conquer for efficiency and our scheme's modulus size (bits) is 1024 bits.

The *SHRSA messaging scheme's decryption algorithm* is shown below:

The RSA modulus was modified so that it can further decrease the decryption time. It consists of k primes $p_1, p_2, \ldots, p_k$ instead of using only two.

For compute we have consider k=3,

1. Calculate $d_p = d \bmod p\text{-}1$, $d_q = d \bmod q - 1$ and $d_r = d \bmod r\text{-}1$.
2. Calculate $M_p = C^{d_p} \bmod p$, $M_q = C^{d_q} \bmod q$, $M_r = C^{d_r} \bmod r$.
3. Calculate M from M_p, M_q and M_r using CRT

SHRSA Decryption

$$= k * (n_d + {}^1\!/_{2n_d}) n_p^2$$

$$= k * \left({}^n\!/_k + {}^k\!/_{2n}\right) \left({}^n\!/_k\right)^2 = \frac{n^3}{k^2} + \frac{n}{2} \approx {}^1\!/_{k^2}({}^1\!/_{n^3})$$

We have considered k=3 (N=pqr), so it almost 9 times faster.

We got *decryption complexity* as shown in Equation 2-

$$\text{Decryption complexity}_{(\text{SHRSA})} = \left(3 * \left(n - {}^n\!/_3\right)\left({}^n\!/_3 + 2\right)\right) + \left(3 * \left(3 * \left({}^n\!/_3\right)^3 + \left({}^n\!/_3\right)^2\right)\right) + 16{}^{n^2}\!/_3 + o(n^2)$$

$$= {}^{n^3}\!/_3 + 19{}^{n^2}\!/_3 + o(n^2) + 4n \approx {}^{n^3}\!/_3 + o(n^2) \quad (2)$$

One of the advantages of the SHRSA decryption is time, by use of the Chinese Remainder Theorem and doing the calculations in parallel, the number of bit operations need to decrypt a ciphertext is at most

$$\frac{3}{2r^3} \left(\log_2 N\right)^3 \text{ (Using standard arithmetic)}$$

For 1024 bit SHRSA, k = 3 is the case we have used, which offers a theoretical speedup of 2.25 comparison to the standard CRT- RSA decryption. The RSA, CRT-RSA and the SHRSA decryption complexities orders and gain calculations summarized values are shown in Table 3 (n = 1024, k = 3).

The SHRSA messaging scheme's cipher with its SHRSA encryption and decryption has some similarities and differences with Multi-prime RSA and Multi-factor RSA as discussed below.

✓ The SHSRA decryption is with N = 1024 Bits, where each prime is approximately 341 bits. This is alike Multi-prime RSA but multi-factor RSA is in form of $N = p^2q$. The SHRSA works with k = 3 (i.e N = pqr) so this way multi-factor RSA is different from the SHRSA.

TABLE 3. Comparisons of the complexity order of decryptions of variants of RSA.

Variant	Complexity Order	Unit Time for Decryption	Asymptotic Decryption speed gain (considering main RSA's speed up is x)-faster - for n=1024 Bit, k=3
RSA without CRT	$O(n^3)$	1	x
RSA with CRT	$2 \cdot O(n/2)^3)$	¼ = 0.25	4x
SHRSA Decryption	$k.O\left((n/k)^3\right)$	1/9 = 0.11	9x (2.25 times than CRT-RSA)

✓ Multi-prime RSA and Multi-factor RSA does not use Efficient RSA in encryption but we are using the Efficient RSA in the SHRSA encryption. So the scheme is having more and more effortlessness for users with high speed by use of Efficient RSA with Extended Euclidean Algorithm than Multi-prime RSA and Multi-factor RSA.

✓ We are using OAEP and random salts in the SHRSA messaging scheme. This is helping us resolving the following scientific problems-

1. Exploitation of multiplicative property (CCA).
2. Exploitation of homomorphic property (MITMA).
3. Short Plaintext Attack.
4. Factorization of RSA modulus n.
5. Algebraic Attacks.
6. The Hastad Attack.
7. Desmedt-Odlyzko Attack.
8. Related Message Attacks.
9. Fixed Pattern RSA Signature Forgery.
10. Two Attacks by Bleichenbacher.

This is totally different from Multi-prime RSA and Multi-factor RSA.

✓ In the SHRSA messaging scheme, we have protection from Sniffing attack and real-time key negotiation between each peer by PFS grade 4 - Older Diffie-Hellman without curves (DHE). This is totally different from Multi-prime RSA and Multi-factor RSA.

✓ The SHRSA scheme's authentication is much stronger than Multi-prime RSA and Multi-factor RSA. Single layer key exchange authentication in Multi-prime RSA and Multi-factor RSA is easy to break these days. So SHRSA's four layered authentication stack in the scheme is resolving following problems-

1. Non-repudiation attack.
2. Man-in-the middle attack.
3. Replay attack.
4. Giving us 3-way handshaking.

5. Both peer authentication by key exchange in every four layers (layer 2 to layer 5).
6. Parallel protection to Sniffing attack.
7. Real-time key negotiation between each peer.

This is totally different from Multi-prime RSA and Multi-factor RSA.

So in a nutshell the SHSRA has stronger security, efficiency and authenticity than Multi-prime RSA and Multi-factor RSA.

IV. SECURITY ANALYSIS OF SHRSA MESSAGING SCHEME'S CIPHER

This section has highlighted the security protections SHRSA messaging scheme's cipher is offering us from several malicious attacks on RSA [1]–[10], [12]–[17], [21]–[26], [29], [38], [43], [44], [49]–[53]. Let's discuss one by one.

A. FACTORIZATION OF RSA MODULUS n

Any attackers if they succeed to get the factorization of n=p.q.r, knowing e, he can easily find the private key d [2], [4], [7], [54]. So it's very clear how much venerable it is. In SHRSA messaging scheme's cipher Monte CarloFactor with Pollard p-1 factorization is being used for shielding the Factorization of RSA modulus n problem of main RSA. The prime number in SHRSA messaging scheme's cipher is decided by the SHRSA Big Integer class with Rabin-Miller Primarily test. Also the salting process is used to enhance the ability for protection from Factorization of RSA modulus n problem.

B. CHOSEN PLAINTEXT ATTACK

Whenever the plaintext space is small, this attack occurs. The attacker encrypts all plaintext messages and then compares which of the cipher texts have matched with the given cipher text c. Similar Short Plaintext Attack can be made when the message is small, although the cipher text can be as big as n. The attacker performs two sets of operations as follows:

$$a = cx^{-e} mod\ n \quad for\ all\ 1 \le x \le 10^9$$
$$b = y^2 mod\ n \quad for\ all\ 1 \le y \le 10^9$$

If for some (x,y), I have a = b then $c = (xy)^e mod\ n$ and hence the plaintext message can be calculated as m = (xy).

These two attacks are shielded in SHRSA messaging scheme's cipher with the *salting process* before encryption, so a large plaintext can be formed. These randomly added digits are removed after decryption. Here *OAEP*is used also for shielding the exploitation of multiplicative property of RSA and exploitation of homomorphic property on RSA. Also another layer of padding by using *PKCS#5* is used for extra protection from this attack.

C. CCA

This attack happens when the receiver signs his message with his private key. The attacker receives the cipher text c, wishes to find the decryption $m = c^d\ mod\ n$ chooses a random integer s and asks receiver to digitally sign the innocent looking message $c' = s^e c\ mod\ n$. From the receiver's answer m′, message $m = \frac{m'}{s}\ mod\ n$ can be recovered by almost no effort. SHRSA messaging scheme's cipher is using one way function to stop the CCA.

D. BROADCAST DECRYPTION BY LOW EXPONENT ATTACK

This attack can be made when the value of the public exponent is small. SHRSA messaging scheme's cipher's m, n and r are large enough so product is again large.

E. BROADCAST DECRYPTION BY COMMON MODULUS ATTACK

If the message m is encrypted twice by use of the RSA system using the public keys $K_1 = (e_1, n)$ *and* $K_2 = (e_2, n)$ with a common modulus n and gcd $(e_1, e_2) = 1$, then n can be quickly recovered as from the cipher texts $c_1 = m^{e_1} mod\ n$ and $c_2 = m^{e_2} mod\ n$ using following procedure-

a. Compute x_1, x_2 satisfying the equation $x_1 e_1 + x_2 e_2 = 1$ by use of the extended Euclidean Algorithm where the indices are chosen such that $x_2 < 0$.

b. Determine y satisfying $1 = yc_2 + kn$ by the extended Euclidean Algorithm.

c. Then the plaintext message can be calculated as $c_1^{x1} y^{-x2}$.

To avoid this attack, with the same modulus and relatively prime encryption exponents, no identical messages are sent to receivers.

F. FAULT INJECTION ATTACK

This attack is typically a malicious attack where the fault in the computation of the private key results into the computation of the private key. In CRT-RSA, the signature generation is consist of two exponentiations $S_p = m^{d_p}$ mod p and $S_q = m^{d_q}$ mod q *where* $d_p = d\ mod\ p-1$ and $d_q = d\ mod\ q-1$. The signature can be easily obtained using the Garner's formula:

$$S = S_q + q(i_q\ (S_p - S_q)\ mod\ p)$$

where $i_q = q^{-1} mod\ p$.

Now if a fault is injected during the calculation of S_p resulting to a faulty signature $\tilde{S}$. Since $S \equiv S_p\ mod\ p$ and $S \equiv S_q\ mod\ q$. But the faulty signature $\tilde{S}$ actually satisfy $\tilde{S} \equiv S\ mod\ q$ and $\tilde{S} \equiv S\ mod\ p$. Therefore the secret parameter q can be easily calculated by computing the gcd of $S - \tilde{S}$ mod and N. The private key can be easily calculated after finding q. We are resolving this attack by checking the signature is returned if only if $S^e\ mod\ n = m$. We have three primes, so we have implemented this for three primes.

G. SMALL DIFFERENCE BETWEEN PRIMES p AND q ATTACK

If the prime numbers p and q are close to each other, an efficient factorization of the modulus can give us the values of the prime numbers and hence the private key.

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2$$

For the factorization of n, we must check exactly all integers $> \sqrt{n}$, for which $x^2 - n$ is a perfect square. Then the primes p and q can be obtained as x+y and x−y. The primes in SHRSA messaging scheme's cipher's Big integer class are already maintaining the logic of large differences to avoid this attack.

H. FINDING THE e^{th} ROOT ATTACK

This attack computes the plaintext by finding the e^{th} root of the cipher text. Here, the plaintext m is computed as

$$m \equiv \sqrt[e]{c} \, mod \, n$$

Finding the e^{th} root is a tough, if n is large. But $\varphi(n)$ is given, it can be found in polynomial time. We are shielding this attack with e = 3 with CRT.

I. COMMON PRIME ATTACK

This attack happens when the sender uses (n, e) and (n', e') as public moduli and gcd $(n, n') = p$. The SHRSA prime number generator class is used for shielding this attack.

J. EXPLOITATION OF MULTIPLICATIVE PROPERTY PROBLEM OF RSA

The OAEP added with some random salt are used to get shield from this attack.

K. EXPLOITATION OF HOMOMORPHIC PROPERTY

The OAEP added with some random salt are used to get shield from this attack.

L. INTEGER FACTORIZATION PROBLEM AND PROBLEM OF LARGE PRIVATE EXPONENTS

We are getting shielding from these attacks with 1024-bit RSA based SHRSA big Integer class and we are having the customized Monte Carlo Factor with Pollard p-1 factorization for this and our prime number is generated by the SHRSA Big Integer class with Rabin-Miller Primarily test.

M. RSA EXPONENT AND EFFICIENCY PROBLEM AND RSA PRIVATE EXPONENT PROBLEM

We are getting shielding from this problem by using SHRSA Big Integer class. We are having the customized Monte Carlo Factor with Pollard p-1 factorization for this and our prime number is generated by the SHRSA Big Integer class with Rabin-Miller Primarily test. Also PKCS #5 is used for defining the exponents as inverses $\lambda(N) = \text{lcm}\ (x - 1, y - 1)$. Also before encryption starts in SHRSA's messaging scheme's cipher, the OEAP insertion with random salts in the 9 layered SHRSA cipher is shielding the following attacks -Algebraic attacks, Hastad attack, Desmedt-Odlyzko attack, Related Message attacks, Fixed Pattern RSA Signature Forgery and two attacks by Bleichenbacher.

Also SHRSA scheme is *Set Partial Domain One.*

By use of PFS grade 4 - Older Diffie-Hellman without curves (DHE) [20], [24] in SHRSA's messaging scheme's cipher, the cipher is shielded from following attacks -non-repudiation attack, MITMA, CCA and replay attack.

V. PERFORMANCE EVOLUTION AND ANALYSIS

The SHRSA messaging scheme is designed in the Java and we are using NetBeans IDE 8.2. For SHRSA encryption and SHRSA decryption we have 21 classes for each one, as a whole 42 classes. We have described our details software package in our last work [46], [48]. In our past works [46], [48], we have already tested with n number of SHRSA clients and n number of SHRSA servers trying to communicate with each other as peer. But we have shown that at a time only one peer can start secure E2E authenticated messaging and other all peers will be connected to each other but has to wait for present communicating peer to end first.

We have first checked the memory performance of the SHRSA messaging scheme's SHRSA server and client with RSA server and client with the setup desktop with Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz (8 CPUs), ~4.0GHz. For testing comparison purpose, we have developed the API for RSA server and client messaging. Already we have our APIs for SHRSA server and client with 9 layered secure communication protocol stack.

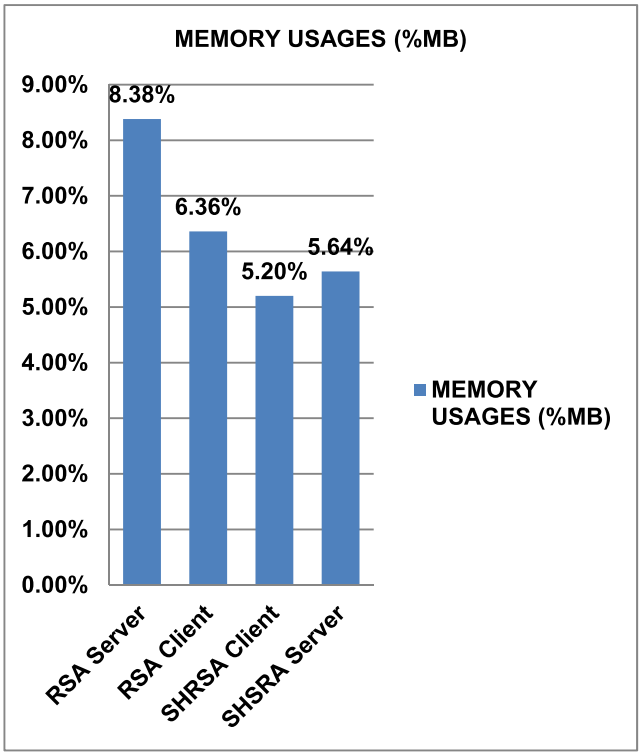


FIGURE 3. RSA with SHRSA Client and SHRSA server's memory usage comparisons.

Now here we have discussed the memory occupancy testing for main RSA server and client messaging API and SHRSA client and server API. Our allocated memory for testing purpose was 512 MB. Now we have run all these four APIs and did constant messaging vice versa for two minutes and have run them 5 times and then did the average. Figure 3 is representing 5 times running average of memory occupancies. The *memory occupancy data (in %)* what we got is shown in below Figure 3. Here we have found that the

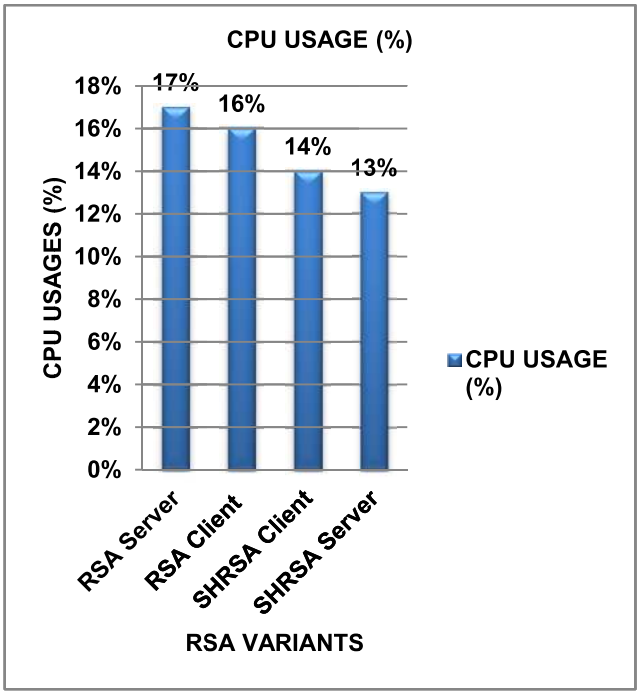


FIGURE 4. RSA with SHRSA Client's and SHRSA server's CPU usages comparisons.

SHRSA messaging scheme's server and client APIs at the time of messaging are *very less memory occupier* (1%-3% less memory occupier) than the main RSA cipher. Details results calculations are in *Appendix A*.

Then during this 2 minutes continuous messaging testing period, we have tested the CPU usages of the SHRSA server and client and main RSA server API and client API. We have run all these four APIs and did constant messaging vice versa for two minutes and have run them 5 times and then did the average. We have found that again the SHRSA client and server is much less CPU occupier than main RSA instant messaging server and client. The CPU occupancy data (in %) what we got is shown in below Figure 4. Here we have found that the SHRSA cipher as a whole is *very less CPU occupier (2%-4% less)* than the main RSA cipher.

Now if we compare the SHRSA cipher's encryption complexity and decryption complexity with main RSA and CRT-RSA, it can be shown in Table 4. The RSA and CRT-RSA decryption complexity calculations are as follows and encryptions are alike as shown in section III-

TABLE 4. Comparisons of SHRSA cipher's encryption complexity and decryption complexity with main RSA and CRT-RSA.

Comparison	RSA	CRT RSA	SHRSA
Encryption Complexity	$(3n_e - 2(n2+n)$	$(3n_e - 2(n2+n)$	$(3n_e - 2)(n^2 + n)$
Decryption Complexity	$3n^3 + o(n^2)$	$\frac{3n^3}{4} + o(n^2)$	$\frac{n^3}{3} + o(n^2)$

RSA

Decryption Complexity $= (3n-2)(n^2+n) \approx 3n^3 + o(n^2)$

CRT-RSA

Complexity of decryption Algorithm $\approx 3n^3/4 + o(n^2)$

Details of calculation of SHRSA cipher's decryption complexity is shown in section III.

Decryption complexity$_{(SHRSA)} = {}^{n^3}/_3 + 19{}^{n^2}/_3 + o(n^2) + 4n \approx {}^{n^3}/_3 + o(n^2)$

So from Figure 3 and Figure 4 and Table 3 and 4, it is very clear that the SHRSA messaging scheme's cipher is *very less memory occupier, very less CPU occupier*and we gain *9 times in decryption* than main RSA theoretically.

Now lets us consider the encryption complexity of the SHRSA scheme, main RSA and CRT-RSA in our case where key size 1024 bit, it is shown in Table 5 (here we have only consider n = 1024 bits as our scheme is based on that). So RSA, CRT-RSA and SHRSA have same complexity due to no changes in the algorithms like main RSA.

TABLE 5. Comparisons of SHRSA cipher's encryption complexity with main RSA and CRT-RSA.

Variants	Encryption Complexity	
RSA	n_e = 16 bits	46 n^2
RSA CRT	n_e = 16 bits	46 n^2
SHRSA	n_e = 16 bits	46 n^2

Following parameters are used:

- n = No of bits in modulus.
- n_e = No of bits in public exponent (e).
- n_d = No of bits in private exponent (d).

Now Here in our research n = 1024, n_e = 16 Bits

So we can calculate as below-

Encryption Complexity $= (3n_e - 2)\,(n^2 + n)$

It can be written $((3 * 16) - 2)n^2 \approx 46n^2$ [Putting value of n_e]

Table 5 is generated as per this.

So it's vivid that RSA, CRT-RSA, and the SHRSA encryption all are having same encryption complexity.

But our scheme's decryption is increasing the computational speed for decryption. Now we have compared the decryption complexity of our scheme and main RSA and CRT-RSA, it is shown in Table 6 (here we have only consider n = 1024 bits and k = 3 as our scheme is based on that).

TABLE 6. Comparisons of SHRSA cipher's decryption complexity with main RSA and CRT-RSA.

Variants	Decryption Complexity	
RSA	n_d = 1024 bits	3073 n^2
RSA CRT	n_d = 512 bits	1543 n^2
SHRSA	n_d = 342 bits	348 n^2

Details calculations are as below-

RSA

- n = No of bits in modulus.
- n_e = No of bits in public exponent (e).
- n_d = No of bits in private exponent (d).

We are using a method to perform modular exponentiations is the so-called *square-and-multiply algorithm*. In its binary

version, the algorithm processes a single bit of the exponent b at a time and on every iteration squares its intermediate result and multiplies it with a if the current bit is set. Thus, the algorithm always performs $t-1$ modular squarings and at most $t-1$ modular multiplications. Since an upper bound on a single modular multiplication – and therefore also squaring – is $O(v^2)$, the repeated square-and multiply algorithm has a running time of $O(tv^2)$ where t is the bit length of the exponent b and v is the bit length of the modulus c.

Given this bound, the encryption exponent e in the original RSA cryptosystem is typically chosen to be a small number, often $2^{16}+1$. There are two reasons for this: the relative short bit length of $2^{16}+1$ will result in a small amount of modular squarings, and $2^{16}+1$ has only two 1's in its binary representation leading to the fewest possible modular multiplications for valid RSA encryption exponents. Expressed informally, choosing e this small almost effectively yields an encryption running time dependent only on the bitlength n of the modulus N, i.e. $O(n^2)$. The structure of the decryption exponent d cannot be tailored to fit the repeated square-and-multiply algorithm in the same way, but will often be long and consist of a random number of 1's in its binary representation. The worst case scenario is that $|d| \approx |N|$ yielding a running time of $O(n^3)$. This means that encryption is much faster than decryption in the original RSA.

So we are considering n = 1024 bit for RSA, so n_d = 1024 bits

So the order of decryption complexity is $O(n^3)$.

Decryption complexity of RSA is $= 3n^3 + n^2 + o(n^2) \approx 3n^3 + o(n^2)$

It can be written as $\approx (3*1024)+1 = 3073n^2$ (n = 1024)

[Here n = 1024, n_e = 16 Bits, n_d = 1024 Bits]

CRT-RSA

Here n_d = 512 bits as it divides the key and calculate like divide and conquer - Calculate $d_p = d \bmod p-1$ and $d_q = d \bmod q-1$. But here CRT-RSA works with two primes number.

Complexity of Decryption Algorithm $\approx 3n^2/4 + 7n^2/2 + o(n^2) \approx \frac{3n^3}{4} + o(n^2)$

It can be written as $\approx (3/4(1024) + 7/2)\, n^2 \approx (768 + 7/2)\, n^2 \approx 1543n^2$

[Here n = 1024, n_e = 16 Bits, n_d = 512 Bits, we know that Unit time of Decryption is 1 for 1024 bit]

SHRSA

n_d = 342 bits as it works with multi –primes – 3 prime numbers.

$$\text{Decryption complexity}_{(\text{SHRSA})} = \left(3*\left(n-{}^{n}/_{3}\right)\left({}^{n}/_{3}+2\right)\right)+\left(3*\left(3*\left({}^{n}/_{3}\right)^{3}+\left({}^{n}/_{3}\right)^{2}\right)\right) +16{}^{n^2}/_{3}+o(n^2) = {}^{n^3}/_{3}+19{}^{n^2}/_{3}+o(n^2)+4n \approx {}^{n^3}/_{3}+o(n^2)$$

It can be written as $\approx (1/3(1024)+19/3) = (1024+19)/3 = 347.6666\ldots n^2 = 348n^2$

[Here n = 1024, n_e = 16 Bits, n_d = 342 Bits, we know that Unit time of decryption is 1 for 1024 bit]

Now in real-time testing, we have found the running time for encryption and decryption of the each scheme (RSA, CRT-RSA, SHRSA) as shown in Figure 5. These encryption and decryption times are in milliseconds and the result times are shown in Figure 5, these are the average times of 8 times running of encryption and decryption of each of API of RSA, CRT-RSA and SHSRSA. So it's clear that the SHRSA decryption running time is gaining *2.248 times* than CRT-RSA and also it is gaining *8.858 times* than main RSA.

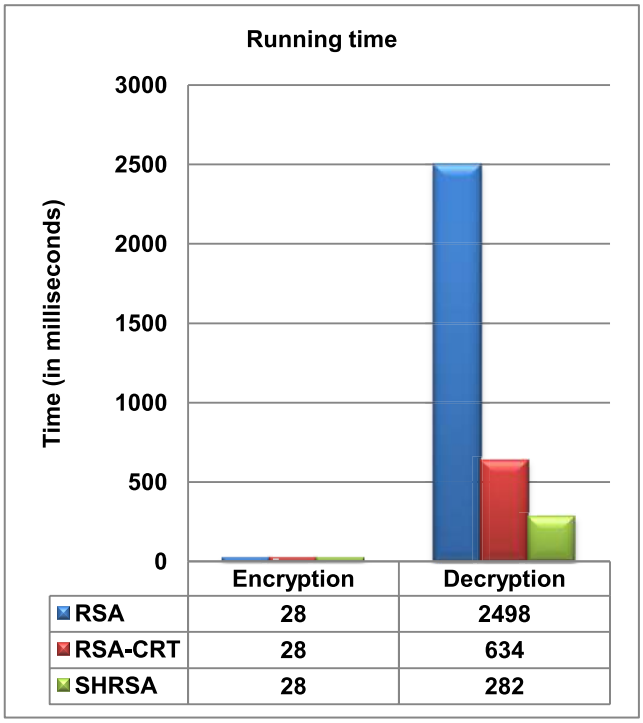


FIGURE 5. Running time comparisons of SHRSA and main RSA and CRT-RSA.

TABLE 7. Comparisons of RSA, CRT-RSA and SHRSA cipher's encryption execution time with different text sizes.

Input Size of Text (KB)	Encryption Execution Time (Seconds)		
	RSA	CRT-RSA	SHRSA
0.150	0.0245..	0.02811	0.02825
0.165	0.0260..	0.02829	0.02841
0.170	0.0279..	0.02842	0.02856
0.185	0.0291..	0.02859	0.02865
0.205	0.0301..	0.02871	0.02879

Now we have performed encryption with RSA, CRT-RSA and SHRSA messaging scheme with different text sizes (0.158 KB, 0.159 KB, 0.160 KB, 0.161 KB, 0.162 KB) as per our text box limit and calculated their throughput. The summarized results of encryption execution time (in seconds) with corresponding text file size are in Table 7. We have found that RSA, CRT-RSA and SHRSA's encryption throughput is alike, its equivalent to 6 KB/Sec. Details calculations are

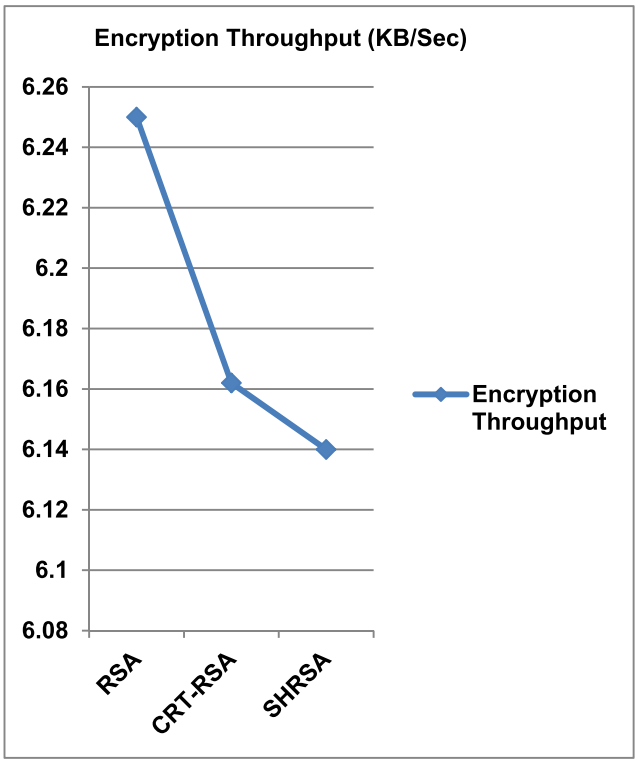


FIGURE 6. Comparisons of RSA, CRT-RSA and SHRSA's encryption throughput.

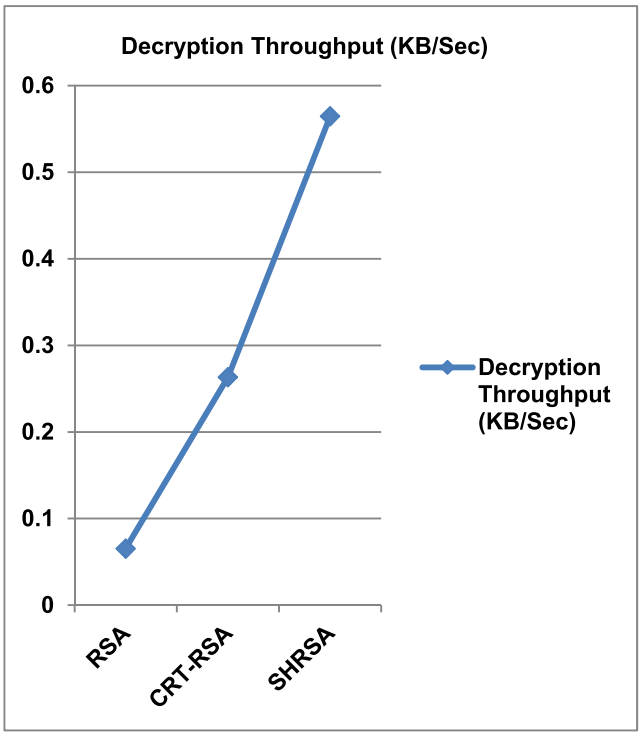


FIGURE 7. Comparisons of RSA, CRT-RSA and SHRSA's decryption throughput.

in Appendix B. Still micro level exact value comparisons of RSA, CRT-RSA and SHRSA's encryption throughput are shown in Figure 6.

Now we have performed decryption with RSA, CRT-RSA and SHRSA messaging scheme with different text sizes (0.158 KB, 0.159 KB, 0.160 KB, 0.161 KB, 0.162 KB) as per our text box limit and calculated their throughput. The summarized results of decryption execution time (in seconds) with corresponding text file size are in Table 8. Figure 7 is showing the comparisons of RSA, CRT-RSA and SHRSA's decryption throughput. Details calculations are in Appendix C.

TABLE 8. Comparisons of RSA, CRT-RSA and SHRSA cipher's decryption execution time with different text sizes.

Input Size of Text (KB)	Decryption Execution Time (Seconds)		
	RSA	CRT-RSA	SHRSA
0.150	2.6765	0.6615	0.274
0.165	2.6792	0.6634	0.298
0.170	2.6805	0.6652	0.311
0.185	2.6814	0.6673	0.334
0.205	2.6828	0.6691	0.351

We have found that RSA, CRT-RSA and SHRSA's encryption throughput are alike (all are around 6 KB/Sec) but decryption throughputs are distinguishable (Figure 6 and Figure 7). Still micro level exact value comparisons of RSA, CRT-RSA and SHRSA's encryption throughput and decryption throughput are shown in Figure 8. It is clear

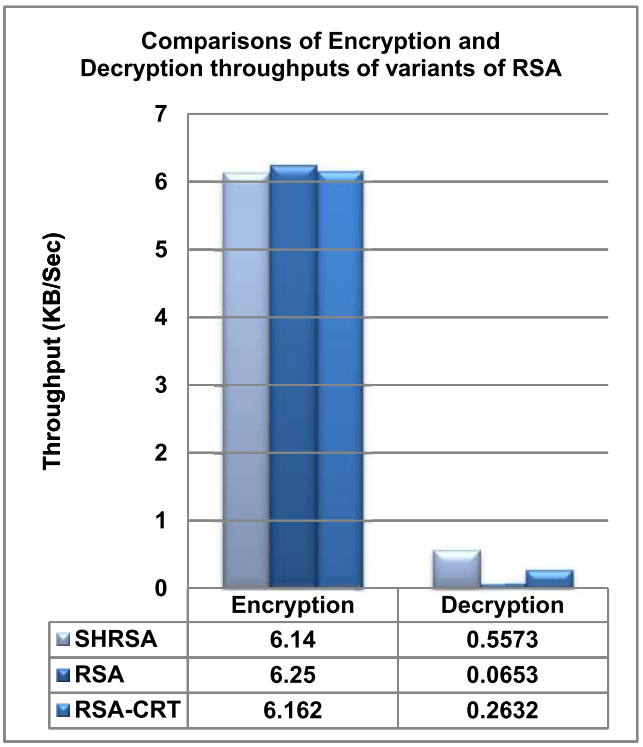


FIGURE 8. Comparisons of encryption and decryption throughputs of RSA, CRT-RSA and SHRSA.

from Figure 8 by decryption throughput SHRSA has gained 8.5345 times than main RSA (0.5573/0.0653≈8.5345) and it also has gained 2.1174 (0.5573/0.2632≈2.1174) times

than CRT-RSA. So though *theoretically the SHRSA messaging scheme's decryption gain is 9 times compared to main RSA but practically it is gaining 8.858 times than main RSA (Figure 5) and has almost 8.5 times higher decryption throughput than RSA and 2.1 times higher than CRT-RSA.*

So these *less memory occupancy, less CPU occupancy, then gaining in decryption time and decryption throughput have made the SHRSA scheme very relevant to these days Internet, CPS and IoE 's need.*

VI. THE MAJOR CONTRIBUTIONS OF THE WORK AND SHORTCOMINGS OF THE WORK

So at the beginning of discussing the major contributions of the work, lets discuss the encryption level contribution, the SHRSA messaging scheme's 9 layered cipher's encryption with 1024 Bit RSA modulus, is shielding us from some of the scientific problems of RSA like, *the very high computationally costly exponentiation modulo N problem, the exploitation of multiplicative property, low modular complexity with effortlessness, difficulty of the integer factorization problem of RSA, the exploitation of homomorphic property and speediness problem.*We all know that all available RSA variants' encryption are able to solve two or three major problems of RSA but the SHRSA messaging scheme's encryption is resolving many problems of RSA as discussed in section III and section IV in details.

Also the SHRSA messaging scheme's 9 layered cipher's encryption has proper protection from *CCA and Short Plaintext Attack etc, along with protection to Sniffing attack* and *the real-time Key negotiation issue also. Brute force attack is shielded by randomly changing the keys in synchronous time slot with 1024 Bit value.*

In the decryption process, the SHRSA messaging scheme's 9 layered cipher's decryption is shielding us from some of the scientific problems of RSA like, *asymptotic very low speed of decryption of RSA problem, computational modular exponentiation complexity and partial key exposure vulnerability*. We are getting almost *8.858 times* faster asymptotic decryption speed practically than RSA.

The *SHRSA messaging scheme with 9 layered cipher* is able to replace the existing disadvantages of existing *IM schemes* and *protocols* and giving us following advantages-

1. It's distributed system, no single point failure with SHRSA and its peer to peer nature for all kind of users.
2. SHRSA Decryption is gaining *8.858 times* than RSA.
3. SHRSA Decryption has almost *8.5 times higher* decryption throughput than RSA.
4. SHRSA encryption and decryption is much more complex between each peer.
5. OEAP with some random salts added on runtime with synchronize time gap protects from CCA and Short Plaintext Attack, MITMA and other attacks.
6. It works with any network with dual stack with native IPv4.
7. E2E encryption with full mesh topology.
8. No default setting shared with others.
9. No need of any third party.
10. It's more reliable, more efficient and stronger due to variants of RSA integration.
11. No need to install IMSecure.
12. We have our own four layered authentication stack in the cipher.
13. No need of use of any password as we have our own four layered authentication stack for peer.

Also as a whole, the SHRSA messaging scheme with 9 layered cipher is *lightweight, efficient* and *providing strong authentication with 4 layered stacked authentication structure*. The lightweight and efficiency feature will make the SHRSA cipher more relevant to be integrated in IoT and CPS structures in future. Also it's able to provide pure peer to peer strong and reliable encryption eliminating several attacks. As it is peer to peer nature, so for VPN and LAN to LAN tunnel sessions, it is an ideal scheme.

Now the SHRSA messaging scheme's 9 layered cipher can be used in following real-time daily use-

- In a distributed peer to peer communication environs this cipher can be used.
- CPS, IoT and IoE's secure communication protocols should be E2E encrypted, lightweight and efficient and SHRSA messaging scheme's cipher have similar features, so it is promising cipher to be integrated in IoT and IoE architectures in near future for secure communication.
- Replacement of SSL/TLS where needed for personal messaging scenario, the SHRSA scheme can be used (as SSL/TLS has several backlogs).
- The SHRSA's 9 layered cipher can be incorporated in the Future Internet Architectures.
- We have the 4 layered authentication stack for peers and then SHRSA encryption, so in a scenario where there is a need of use of any password (like-AOL Instant Messenger (AIM), ICQ, MSN Messenger (Windows Messenger in XP), and Yahoo! Instant Messenger (YIM)) the SHRSA can be used without any password.

Now let's highlights some *similarity and dissimilarities of SHRSA messaging scheme's cipher's working principles and Blockchain's consensus protocols' working principles*. Those are shown in Table 9. We all know that in future Blockchain technology can be used in IoT in various micro areas.

The shortcomings of the SHRSA messaging scheme are as follows-

1. It works over TCP. So IP address identified with TTP system, so identification of the peer is disclosed.
2. As we have multiple techniques applied specially padding like OAEP with PKCS #5 with salts in SHRSA for security reasons so text size for this messaging scheme has limitations in bytes- block size must be between 1 and 255 bytes.
3. Now this SHRSA messaging scheme's cipher is used in IM purpose, it's not a broad use. Future usages can be

TABLE 9. Comparisons of similarity and dissimilarities of SHRSA messaging scheme's cipher's working principles and Blockchain's consensus protocols' working principles.

Working principles	Blockchain's consensus protocols	SHRSA messaging scheme's cipher
Similarities	1. Works pure peer to peer. 2. Cryptographic approach used for security- SHA 256. 3. Distributed system, no central point failure. 4. Blockchain can be used for security applications for the IoT. Examples can be- a. decentralized applications enabling the smart objects to interact with security, b. form payments mechanisms, c. form PKI services d. execute Multiple Secure Computation (MPC) e. provide support for Smart Ambient , and give privacy in storage systems	1. Works pure peer to peer. 2. Cryptographic approach used for security-1024 bit SHRSA cipher. 3. Distributed system, no central point failure. N number of servers and N number of clients can work together. 4. Efficiency and lightweight features are promising for future cryptographic use. As SHRSA cipher is also peer to peer crypto approach so can do these functions (a to e points of Blockchain's consensus protocols) also.
Dissimilarities	1. All blocks are anonymous, no disclose of identity. 2. Scalability is a big problem like we have 51% attacks. 3. Efficiency is still a big problem. 4. It is not based on TTP. 5. It uses centralized services: users, for various causes, do not save and control their private keys, delegating this function to outsourced services. 6. Various practical domains based on Blockchain provide decentralized security and privacy but with the cost of excessive consumption of energy and delays.	1. All clients are identified by IP address and mirror the IP address. 2. Scalability is not a problem. 3. In similar type of RSA based approaches it is efficient. 4. SHRSA is based on TTP. 5. SHRSA cipher's every client and server work pure peer to peer in a decentralized way. 6. SHRSA cipher's present practical domain is secure, and authenticated messaging which is efficient and consumes less memory and CPU usages. Future domains are experimental.

in CPS, IoE and Blockchain for E2E secure, efficient and lightweight communication.

4. To resolve many scientific problems of RSA we have used many existing approaches and made them Hybrid, so in future as per application specific, the decryption throughput can be improved further.
5. The lightweight modularity can be further improved.

CONCLUSIONS

The efficient and four-layered authenticated SHRSA messaging scheme is resolving the issue of low modular complexity of RSA and is able to provide effortlessness and speediness features to users. It offers parallel protection to Sniffing attack with benefits of real-time key negotiation between each peer by 4 layered authentication stack powered by PFS grade 4 - Older Diffie-Hellman without curves (DHE). Exploitation of multiplicative property and homomorphic property of RSA are defended by the scheme. With 1024 bit keys we are resolving the difficulty of the integer factorization problem of RSA. The Square and multiply algorithm has helped us to resolve the very high computationally costly exponentiation modulo N problem of RSA. The scheme is able to battle the Brute force attack by randomly changing the keys in synchronous time gap with 1024-bit value. The SHRSA decryption provides us solution for computational modular exponentiation complexity of RSA and partial key exposure vulnerability of RSA. We can use our scheme's cipher in several real-time environs like CPS, IoT and IoE for E2E secure communication. Any desktop and any laptop connected in VPN or LAN can be the peer device for using this scheme. This scheme can be used in a distributed environment, where multiple servers and multiple clients can communicate in a peer to peer manner with strong, reliable and efficient E2E security. User with need of very strong authentication can use this, as our scheme is equipped with 4 layered authentication stack. This scheme can be a replacement of SSL/TLS where needed for personal messaging scenario. The architecture also affords a hassle-free, secure, peer-to-peer, unconventionally strong and reliable platform with E2E encryption for people and organizations who are concerned about their privacy and security. We have our own authentications for peer and then SHRSA encryption, so in a scenario where there is a need of use of any password (like-AOL Instant Messenger (AIM), ICQ, MSN Messenger (Windows Messenger in XP), and Yahoo! Instant Messenger (YIM)) our scheme can be used without password. In a scenario where external digital certificates are used, our scheme can work without external digital certificates, as we have SHRSA's security, authentication and highly efficient architecture with very strong ability to maintain CIA triad. In a scenario, where there is need of any third party (like Instant Messaging Key Exchange (IMKE) protocol), the SHRSA scheme can work well, as no need of third party authentication. This scheme has replaced the disadvantages of IPSec, so for personal messaging we can use this secured scheme without IPSec. We have found that in evolutions and analysis of the scheme, it is not only resolving various scientific problems of RSA but also occupying 2%-4% less CPU than main RSA and occupying 1%-3% less memory than main RSA. Its decryption average time and throughput are also significantly better than RSA and CRT-RSA. The results obtained have shown us that the scheme is lightweight with high security. So as a whole these features of the SHRSA scheme have made it very relative to be integrated in IoT,

IoE and CPS systems for E2E secure communications in future.

APPENDIX A
RSA WITH SHRSA CLIENT AND SHRSA SERVER'S MEMORY USAGE COMPARISONS

We are getting % of memory usage (average of 5 times running) like this, for example SHRSA client memory usage was 26.60 MB out of 512 MB available during our testing. So usage percentage is – $(26.60/512)*100 = 5.1953\%$

RSA Variants	Memory usage percentage	Available memory (MB)	Occupied memory (MB)
RSA Server	8.38%	512	42.90
RSA Client	6.36%	512	32.51
SHRSA Client	5.20%	512	26.60
SHSRA Server	5.64%	512	28.88

APPENDIX B
ENCRYPTION THROUGHPUT CALCULATIONS OF RSA, CRT-RSA, SHRSA

Input Size of Text (KB)	Encryption Execution Time (Seconds)		
	RSA	CRT-RSA	SHRSA
0.150	0.0245..	0.02811	0.02825
0.165	0.0260..	0.02829	0.02841
0.170	0.0279..	0.02842	0.02856
0.185	0.0291..	0.02859	0.02865
0.205	0.0301..	0.02871	0.02879

RSA Encryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time}$$
$$= \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(0.0245+0.0260+0.0279+0.0291+0.0301)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(0.1376)}{5}} = \frac{0.175}{0.02752}$$
$$\approx \frac{0.175}{0.028} = 6.25\ \text{KB/Sec}$$

CRT-RSA Encryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time}$$
$$= \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(0.02811+0.02829+0.02842+0.02859+0.02871)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(0.14212)}{5}} = \frac{0.175}{0.028424}$$
$$\approx \frac{0.175}{0.0284} = 6.1619718\ldots. \approx 6.162\ \text{KB/Sec}$$

SHRSA Encryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time}$$
$$= \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(0.02825+0.02841+0.02856+0.02865+0.02879)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(0.14266)}{5}} = \frac{0.175}{0.028532}$$
$$\approx \frac{0.175}{0.0285} \approx 6.140\ \text{KB/Sec}$$

APPENDIX C
DECRYPTION THROUGHPUT CALCULATIONS OF RSA, CRT-RSA, SHRSA

Input Size of Text (KB)	Decryption Execution Time (Seconds)		
	RSA	CRT-RSA	SHRSA
0.150	2.6765	0.6615	0.274
0.165	2.6792	0.6634	0.298
0.170	2.6805	0.6652	0.311
0.185	2.6814	0.6673	0.334
0.205	2.6828	0.6691	0.351

RSA Decryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time} \times \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(2.6765+2.6792+2.6805+2.6814+2.6828)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(13.4004)}{5}} = \frac{0.175}{2.68008}$$
$$\approx \frac{0.175}{2.68} = 0.065298\ldots \approx 0.0653\ \text{KB/Sec}$$

CRT-RSA Decryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time}$$
$$= \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(0.6615+0.6634+0.6652+0.6673+0.6691)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(3.3265)}{5}} = \frac{0.175}{0.6653}$$
$$\approx \frac{0.175}{0.665} = 0.2631578\ldots \approx 0.2632\ \text{KB/Sec}$$

SHRSA Decryption Throughput (KB/Sec)

$$= \frac{\sum Input\ Files}{\sum Encryption\ execution\ time}$$
$$= \frac{\left(\frac{(0.150+0.165+0.170+0.185+0.205)}{5}\right)}{\frac{(0.274+0.298+0.311+0.334+0.351)}{5}}$$
$$= \frac{\left(\frac{(0.875)}{5}\right)}{\frac{(1.568)}{5}} = \frac{0.175}{0.3136}$$
$$\approx \frac{0.175}{0.314} = 0.5573248\ldots \approx 0.5573\ \text{KB/Sec}$$

ACKNOWLEDGMENT

The authors would like to convey their gratitude and tribute to Late Prof. Wang Jing for his constant supervision and encouragement for this project.

REFERENCES

[1] W. Diffie and Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Aug. 1967. doi: 10.1109/TIT.1976.1055638.

[2] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, Dec. 1982. [Online]. Available: https://www.math.leidenuniv.nl/ hwl/PUBLICATIONS/1982f/art.pdf

[3] G. I. Davida, "Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem," University of Wisconsin, Milwaukee, WI, USA, Tech. Rep. TR-CS-82-2, Oct. 1982. [Online]. Available: http://www.uwm.edu/ davida/papers/chosen/

[4] H. W. Lenstra, "Factoring integers with elliptic curves," *Ann. Math.*, vol. 126, no. 3, pp. 649–673, Nov. 1987. [Online]. Available: https://pdfs.semanticscholar.org/307a/b08c3d4f551019297d2480597c614af8069c.pdf

[5] M. J. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Trans. Inf. Theory*, vol. 36, no. 3, pp. 553–559, May 1990. [Online]. Available: https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/krypto2ss08/shortsecretexponents.pdf

[6] D. Coppersmith, "Finding a small root of a univariate modular equation," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 96. Springer, 1996, pp. 155–165. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F3-540-68339-9_14.pdf

[7] D. Boneh and R. Venkatesan, "Breaking RSA may not be equivalent to factoring," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 98, Springer, 1099, pp. 59–71. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2FBFb0054117.pdf

[8] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1339–1349, Jul. 2000. [Online]. Available: http://crypto.stanford.edu/dabo/abstracts/lowRSAexp.html

[9] D. Boneh, A. Joux, and P. Q. Nguyen, "Why textbook ElGamal and RSA encryption are insecure," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science). Springer, 2000, pp. 30–44. [Online]. Available: https://link.springer.com/chapter/10.1007%2F3-540-44448-3_3

[10] D. Boneh and H. Shacham, "Fast Variants of RSA," in *CryptoBytes*, vol. 1, no. 5, pp. 1–9, 2002. [Online]. Available: https://hovav.net/ucsd/dist/survey.pdf

[11] D. A. Wagner, "Cryptanalysis of a provably secure CRT-RSA algorithm," in *Proc. 11th ACM Conf. Comput. Commun. Secur.*, 2004, pp. 92–97. [Online]. Available: https://people.eecs.berkeley.edu/ daw/papers/crtrsa-ccs04.pdf

[12] M. J. Hinek, "Lattice attacks in cryptography: A partial overview," Univ. Waterloo, Waterloo, Canada, Tech. Rep. CACR 2004-08, 2004. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2004/cacr2004-08.pdf

[13] M. J. Hinek, "New partial key exposure attacks on RSA revisited," Univ. Waterloo, Waterloo, Canada, Tech. Rep. CACR 2004-02, 2004. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2004/cacr2004-02.pdf

[14] M. J. Hinek, "(Very) large RSA private exponent vulnerabilities," Univ. Waterloo, Waterloo, Canada, Tech. Rep. CACR 2004-01, 2004. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2004/cacr2004-01.pdf

[15] C.-Y. Chen, C.-Y. Ku, and D. C. Yen, "Cryptanalysis of large RSA exponent by using the LLL algorithm," in *Appl. Math. Comput.*, vol. 169, no. 1, pp. 516–525, Oct. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009630030400791X

[16] Blomer, May, A., "A tool kit for finding small roots of bivariate polynomials over the integers," in *Advances in Cryptology–EUROCRYPT* (Lecture Notes in Computer Science). Springer, 2005, pp. 251–267. [Online]. Available: https://www.iacr.org/cryptodb/archive/2005/EUROCRYPT/1930/1930.pdf

[17] D. R. L. Brown, "Breaking RSA may be as difficult as factoring," *J. Cryptol.*, vol. 29, no. 1, pp. 220–241, Jan. 2016. [Online]. Available: https://eprint.iacr.org/2005/380.pdf

[18] M. J. Hinek, "Small private exponent partial key-exposure attacks on multi-prime RSA," School Comput. Sci., Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/versions?doi=10.1.1.84.8260

[19] H. M. Sun and M. E. Wu, "An approach towards rebalanced RSACRT with short public exponent," Dept. Comput. Sci., National Tsing Hua Univ., Hsinchu, Taiwan, Tech. Rep. 2005/053, 2005. [Online]. Available: https://eprint.iacr.org/2005/053.pdf

[20] *Diffie-Hellman Without Curves (DHE)–Perfect Forward Security' (PFS) Usage*. Accessed: Sep. 2015. [Online]. Available: https://www.rsaconference.com/writable/presentations/file_upload/dsp-r01-_seven-grades-of-perfect-forward-secrecy.pdf

[21] H. L. Nguyen, "RSA threshold cryptography," M.S. thesis, Dept. Comput. Sci., Univ. Bristol, Bristol, U.K., 2005. [Online]. Available: https://www.cs.ox.ac.uk/files/269/Thesis.pdf

[22] M. J. Hinek and D. R. Stinson, "An inequality about factors of multivariate polynomials," Univ. Waterloo, Waterloo, Canada, Tech. Rep. CACR 2006-15, 2006. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2006/cacr2006-15.pdf

[23] M. J. Hinek, "Another look at small RSA exponents," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science). Springer, 2006, pp. 82–98. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F11605805_6.pdf

[24] *IEEE 1363-2000 (Specifications for Public-Key Cryptography)–D.5 Scheme-Specific Considerations (Annex D.5.1)*. Accessed: Apr. 2016. [Online]. Available: http://www.freestd.us/soft/4175.htm and https://infostore.saiglobal.com/en-au/Standards/IEEE-1363-2000-572839_SAIG_IEEE_IEEE_1311095/

[25] M. J. Hinek, "On the security of some variants of RSA," Ph.D dissertation, Dept. Comput. Sci., Univ. Waterloo, Waterloo, ON, Canada, 2007. [Online]. Available: https://uwspace.uwaterloo.ca/bitstream/handle/10012/2988/mjhinek_PhD_thesis_2007.pdf?sequence=1

[26] Alhasib, A, Haque, A.L., "A Comparative Study of the Performance Issues of the AES and RSA Cryptography," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.(ICCIT)*. Busan, South Korea, Aug. 2008, pp. 505–510. [Online]. Available: https://ieeexplore.ieee.org/document/4682291/

[27] C. S. Turner, "Euler's Totient function and public key cryptography," *Nov*, vol. 7, p. 138, Mar. 2008. [Online]. Available: http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/RSAmath.pdf

[28] D. Leusse, P. Periorellis, and P. Dimitrakos, "Self-managed security cell a security model for the future internet architectures and services advances in future internet," in *Proc. 1st Int. Conf. Digit. Object Identifier*, Aug. 2009, pp. 47–52. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5223407

[29] Oleshchuk, V., "Internet of things and privacy preserving technologies," in *Proc. Wireless Commun., Veh. Technol., Inf. Theory Aerosp. Electron. Syst. Technol.* Aalbrg, Denmark, Sep. 2009, pp. 336–340. [Online]. Available: https://ieeexplore.ieee.org/document/5172470/?arnumber=5172470

[30] C. M. Medaglia and T. Serbana, "An overview of privacy and security issues in the internet of things C II the internet of things," in *Proc. 20th Tyrrhenian Workshop Digit. Commun.* New York, NY. USA: Springer, 2010, pp. 389–394. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4419-1674-7_38

[31] R. H. Weber, "Internet of things—New security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, no. 1, pp. 23–30, Jan. 2010. [Online]. Available: https://pdfs.semanticscholar.org/9d65/08a44e957d837490d69936db5a211432a411.pdf

[32] A. Boldyreva, H. Imai, L. Fellow, and K. Kobara, "How to strengthen the security of RSA-OAEP," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 1–5, Nov. 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5605382/

[33] H. M. Bahig, A. Bhery, and D. I. Nassr, "Cryptanalysis of multi-prime RSA with small prime difference," in *Information and Communications Security* (Lecture Notes in Computer Science). Springer, 2012, pp. 33–44. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-642-34129-8_4

[34] M. Brachmann, O. M. Garcia, S. L. Keoh, and S. Kumar, "Security considerations around end-to-end security in the IP-based Internet of things," in *Proc. Workshop Smart Object Secur.*, H. Tschofenig, Ed. Mar. 2012. [Online]. Available: http://www.lix.polytechnique.fr/hipercom/SmartObjectSecurity/papers/SyeLoong.pdf

[35] K. Ma, H. Liang, and K. Wu, "Homomorphism property-based concurrent error detection of RSA: A countermeasure to fault attack," in *IEEE Trans. Comput.*, vol. 61, no. 7, pp. 325–336, Jul. 2012. [Online]. Available: https://ieeexplore.ieee.org/document/5953581/

[36] R. Seggelmann, M. Táxen, and E. Rathgeb, "Strategies to secure end-to-end communication—And their application to SCTP-based communication," Ph.D. dissertation, Fac. Econ. Bus. Admin. Inst. Comput. Sci. Bus. Inf. Syst., Univ. Duisburg-Essen, Duisburg, Germany, Dec. 2012. [Online]. Available: here-http://docplayer.net/340722-Sctp-strategies-to-secure-end-to-end-communication-dissertation.html
[37] H. Zhou, "*The Internet of Things in the Cloud: A Middleware Perspective*. Boca Raton, FL, USA: CRC Press, 2013, p. 365.
[38] F. F. Moghaddam and M. T. O. Alrashdan Karimi, "A hybrid encryption algorithm based on small-e and efficient RSA for cloud computing environments," *J. Adv. Comput. Netw.*, vol. 1, no. 3, pp. 119–194, Sep. 2013. [Online]. Available: http://www.jacn.net/index.php?m=content&c=index&a=show&catid=30&id=59
[39] J. Bradly, J. Barbier, and D. Handler, "Embracing the internet of everything to capture your share of 14.4 trillion," Cisco IBSG, San Jose, CA, USA, White Paper, 2013. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf
[40] A. Takayasu and N. Kunihiro, "Partial key exposure attacks on RSA: Achieving the boneh-durfee bound," in *Selected Areas in Cryptography—SAC*. Springer, 2014, pp. 345–362. [Online]. Available: https://eprint.iacr.org/2018/516.pdf
[41] Balasubramanian, K., "Variants of RSA and their Cryptanalysis," in *Proc. Int. Conf. Commun. Netw. Technol. (ICCNT)*, May 2014, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/7062742/
[42] H. Zhang and T. Takagi, "Improved attacks on multi-prime RSA with small prime difference," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E97, No. 7, pp. 1533–1541, 2014. [Online]. Available: https://kyushu-u.pure.elsevier.com/en/publications/improved-attacks-on-multi-prime-rsa-with-small-prime-difference
[43] N. Somani and D. Mangal, "An Improved RSA Cryptographic System," In *Int. J. Comput. Appl. (0975 - 8887)*, vol. 105, - no. 16, Nov. 2014. [Online]. Available: https://research.ijcaonline.org/volume105/number16/pxc3899820.pdf
[44] K. Balasubramanian, "Variants of RSA and their cryptanalysis," in *Proc. Int. Conf. Commun. Netw. Technol.*, 2014, pp. 145–149. doi: 10.1109/CNT.2014.7062742.
[45] A. Bhattacharjya, X. Zhong, J. Wang, "Strong, efficient and reliable personal messaging peer to peer architecture based on Hybrid RSA," in *Proc. Int. Conf. Internet Things Cloud Comput.* Cambridge, U.K.: The Møller Centre, Mar. 2016, pp. 1–10. [Online]. Available: https://dl.acm.org/citation.cfm?doid=2896387.2896431
[46] A. Bhattacharjya, X. Zhong, J. Wang, "An end to end users two way authenticated double encrypted messaging scheme based on hybrid RSA for the future Internet architectures," *Int. J. Inf. Comput. Secur.*, vol. 10, no. 1, pp. 63–79, 2018. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJICS.2018.089593
[47] A. Bhattacharjya, X. Zhong, J. Wang, and L. Xing, "On mapping of address and port using translation (MAP-T)," *Int. J. Inf. Comput. Secur.*, to be published. [Online]. Available: https://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijics. doi: 10.1504/IJICS.2018.10008372.
[48] A. Bhattacharjya, X. Zhong, and J. Wang "Hybrid RSA-based highly efficient, reliable and strong personal full mesh networked messaging scheme," *Int. J. Inf. Comput. Secur.*, vol. 10, no. 4, pp. 418–36, 2018. doi: 10.1504/IJICS.2018.10010256.
[49] a. Bhattacharjya, X. Zhong, Y. Wang, and X. Li. "Security challenges and concerns of internet of things (IoT)," in *Cyber-Physical Systems: Architecture, Security and Application. EAI/Springer Innovations in Communication and Computing*. S. Guo and D. Zeng, eds. Cham, Switzerland: Springer, 2019.
[50] A. Bhattacharjya, Xi. Zhong, J. Wang, and X. Li, "Secure IoT structural design for smart homes," *Smart Cities Cybersecurity and Privacy*, B. R. Danda and K. Z. Ghafoor, eds. Amsterdam, The Netherlands: Elsevier, 2019, pp. 187–201. doi: 10.1016/B978-0-12-815032-0.00013-5.
[51] A. Bhattacharjya, X. Zhong, J. Wang, and L. Xing, "Digital twin technologies and smart cities," in *Internet of Things Technology, Communications and Computing*. Springer, to be published. [Online]. Available: https://www.springer.com/series/11636?detailsPage=titles
[52] A. Bhattacharjya, X. Zhong, J. Wang, and L. Xing, "CoAP application layer connection-less lightweight protocol for the Internet of Things (IoT) CoAP Security with DTLS Supporting CoAP," in *Internet of Things Technology, Communications and Computing*. Springer, to be published. [Online]. Available: CFP-http://www.wikicfp.com/cfp/1687servlet/event.showcfp?eventid=76834©ownerid=118434
[53] A. Bhattacharjya, X. Zhong, J. Wang, and L. Xing, "Secure hybrid RSA (SHRSA) based multilayered authenticated efficient end to End secure 6-layered personal messaging communication protocol," in *Internet of Things Technology, Communications and Computing*. Springer, to be published. [Online]. Available: CFP-http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=76834©ownerid=118434
[54] R. Wisniewski, and R. Wisniewski, "Representation of primes in the form $p = 6 \times \pm 1$ and its application to the RSA prime factorization," in *Proc. AIP Conf. Proc.*, 2018, Art. no. 080006. doi: 10.1063/1.5079140.

ANIRUDDHA BHATTACHARJYA was born in Kolkata, India. He received the B.Tech. degree in computer science and engineering from the West Bengal University of Technology, and the M.Tech. degree in IT-courseware engineering from Jadavpur University. He is currently pursuing the Ph.D. degree [under Chinese Government Scholarship (CGS)] in Electronic Engineering at Tsinghua University, Beijing, China.

Since 2009, he has been an Academician in various capacities in Computer Science and Engineering Department in India, such as a Lecturer, Assistant Professor, Senior Assistant Professor, and Professor; moreover, he was the Head of the Department in various universities and institutions in India during this period. He has authored 35 papers in international conferences and journals. His research areas include RFID-based architectures and middleware, security in fixed and wireless networks, applications of cryptography, and IoT securities. He has worked voluntarily with over 400 International conferences committees and as the Editorial Board Member of 25 International journals, and a Guest Editor of two Inderscience journals.

Dr. Bhattacharjya is a member of 34 IEEE societies and various IEEE technical committees. He has received the ICDCN 2010 PhD Forum Fellowship, and the Best Paper Award in ACM ICC 2016 at Cambridge University, U.K. He has been an IEEE mentor and ACM Faculty Sponsor, since 2012. He was selected as the Outstanding Reviewer of the *Future Generation Computer Systems Journal*, in 2017.

XIAOFENG ZHONG received the Ph.D. degree in information and communication systems from Tsinghua University, in 2005, where he is currently an Associate Professor with the Department of Electronic Engineering. He has published over 30 papers and holds seven patents. He performs research in the field of mobile networks, including users' behaviors and traffic model analyses, MAC and network protocol design, and resource management optimization.

XING LI received the B.S. degree in radio electronics from Tsinghua University, Beijing, China, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from Drexel University, Philadelphia, PA, USA, in 1985 and 1989, respectively. He is currently a Professor with the Electronic Engineering Department, Tsinghua University. His research activities and interests include statistical signal processing, multimedia communication, and computer networks. He has published more than 300 papers in his research areas. He is a Deputy Director of the China Education and Research Network (CERNET) Center and a Member of the Technical Board of the CERNET Project. He was a Member of Communication Expert Committee of the China National "863" High Technology Project. He is a Formal Chairman of the Asia Pacific Networking Group and a Formal Member of the executive council of the Asia Pacific Network Information Center.