A High Level Implementation of the RSEP Protocol

Suleyman Kondakci

Abstract—**We present a high level implementation of the RSEP protocol developed for security evaluation of remote assets with profiles. RSEP provides an alternative way to security test and evaluation, which mainly consists of a secure communication protocol, back-end services, and a variety of remote evaluation agents. Authentication of such entities is a challenging issue, which requires interoperable, efficient, and easily applicable architecture. Based on emerging standards, implementation of RSEP provides web-service containers consisting of back-end services and security profiles for the assets under evaluation, while the agents perform the remote evaluation.**

***Index Terms*—Secure authentication, security protocol, remote security evaluation.**

I. Introduction

THE Remote Security Evaluation Protocol (RSEP) was first introduced in [1], which consists of a secure communication architecture associated with each asset a security profile, and software services and agents that communicate over the Internet and other open networks. The secure communication architecture uses a secure exchange protocol based on public key cryptography, [2]. Application of RSEP aims at providing seamless, location-neutral, time-neutral, and continuous risk management. It is also aimed at inspiring developers and researchers to develop value-added security evaluation tools, techniques and procedures. RSEP embodies the necessary model to remotely monitor assets, systems, and currently supplied protection mechanisms, whether active or passive, in a timely manner. In particular, secure, timely, precise, and efficient monitoring of critical information systems and infrastructures can be achieved with this approach.

The aim of RSEP is to provide an automated, timely, and efficient evaluation method, which can avoid tedious questionnaires and time-consuming on-site inspection procedures. In addition to the features listed in [1], the RSEP protocol can support risk management in grid networks and pervasive systems. The implementation presented here covers mostly the mutual authentication of communicating ends. Since the scope of this paper is on the implementation of a specific protocol, the reader can refer to [3] and [4] for details on authentication protocols.

Suleyman Kondakci, *Faculty of Computer Sciences, Izmir University of Economics*

Especially, mobile wireless technologies based on General Packet Radio Service (GPRS), Digital RF, WLAN 802.11, and Global Positioning System (GPS) are becoming today's must-have technologies. Because of their popularity, rate of the spread of these systems are becoming sporadically complex. RSEP will also play an important role in support of security evaluation and security provision to the wireless environment. Specifications, application, and design guidelines of RSEP are presented in [1] and evaluation agent in [5]. The requirements, protocol description, methodology, communication, and evaluation procedures of RSEP are discussed in [1], however, here we present a high level implementation of RSEP. Thus, we do not aim to discuss theoretical features and principles of the RSEP protocol.

Ad hoc networks, such as sensor and mobile ad hoc networks are deployed in un-trusted environments. As discussed in [4], authentication is a precursor to any secure interactions in these networks. Recently, numerous authentication protocols have been proposed for ad hoc networks. Due to lack of a common framework to evaluate ad hoc authentication protocols, [4] proposes a generic authentication process and a new taxonomy that clarifies similarities and differences among authentication protocols reported in the literature. A distributed light-weight authentication model for ad hoc networks is presented in [6]. Further, a valuable discussion on scalability problems for flat ad hoc networks can be found in [7]. By exploring the data and entity authentication for hierarchical ad hoc sensor networks, [6] investigates the issue of scalability and self-organizing hierarchical ad hoc architectures. A seamless authentication protocol (SAP) for vertical handoff in heterogeneous wireless networks is presented in [8], where the integration of 802.11 and 3G standards is becoming quite popular while increasing the complexity of authentication techniques. To facilitate such integration, seamless vertical handoff is one of the major challenges because it needs to make physical movement transparent to mobile users and preserves application-level connectivity. As claimed in [8], previous works did not consider the impact of authentication mechanisms on the performance of vertical handoff, especially on its delay. Thus, to reduce this delay, it proposes seamless authentication protocols for vertical handoff in wireless heterogeneous networks.

Every asset considered for evaluation is assigned a Security Profile (SP), [1], containing itemized security attributes of the asset, which denote scored weaknesses or strengths depending on the scope of the evaluation.

The RSEP approach is intended to eliminate the need for ad hoc security scanning while engaging the network owners with routinely and systematical monitoring of their currently implemented security services. In particular, the remote evaluator of RSEP can run evaluation remotely, for example conducting any type of penetration or procedural test, remotely.

The advantage of RSEP, which, primarily based on a remote assessment approach independent of time and location, is obvious. Upon request, tests can be executed wherever and whenever needed. In between the test periods the security administrators are compelled to track incidents and threats and recover from incidents in a timely manner. Continuous control over threats and security breaches can be easily achieved before incidents could occur. This feature is crucial for critical information systems. Obviously, this is also an efficient way of enforcing security policies such that the network owners are being motivated to employ security specialists/administrators. It is quite common for many organizations not to have dedicated security administrators, [9] and [10].

As indicated in [1], all security maintenance and required technical test and evaluation tasks are performed on-site by the network owners or remotely by the RSEP compliant security evaluator. Evaluation results are saved in a security profile (SP) of the asset. In addition to optional remote penetration tests, the remote evaluator has mainly the responsibility for conducting risk assessment against the SP and overall security evaluation tasks that are already carried out by the above-mentioned evaluators. In turn, the necessary operation should be performed to recover from eventual vulnerabilities. Nevertheless, with some restrictions and legislative responsibilities, the remote evaluator can also perform some of the tests; including penetration testing, worm attacks, and denial of service attacks, except the regular security maintenance that is naturally the task of the local staff or the system owner.

II. Overall Architecture

RSEP embodies the necessary model to remotely monitor assets, systems, and currently supplied protection mechanisms, irrespective of whether active or passive, in a timely manner. In particular, secure, timely, and efficient monitoring of critical information systems and infrastructures can be achieved with this approach.

One RSEP requirement is the ability of its application designers to provide transport-independent communication architecture. RSEP is designed fundamentally to utilize a stateful, one-way message exchange paradigm that can run on top of the TCP/IP protocol suite. It is, however, preferable for the implementors to use Web-enabled approaches combined with XML to build RSEP applications and tools.

For the prototype implementation of RSEP agents and services, we have used a Web-enabled approach utilizing the Java Server Pages (JSP) as the back-end SP-service. A lower-level implementation utilizing the connection-oriented (TCP) protocol and socket mechanism can also deliver robust communication architecture. Fig. 1 shows the sequence of operations required to establish a secure interaction between the EA and TOE using the socket based approach.

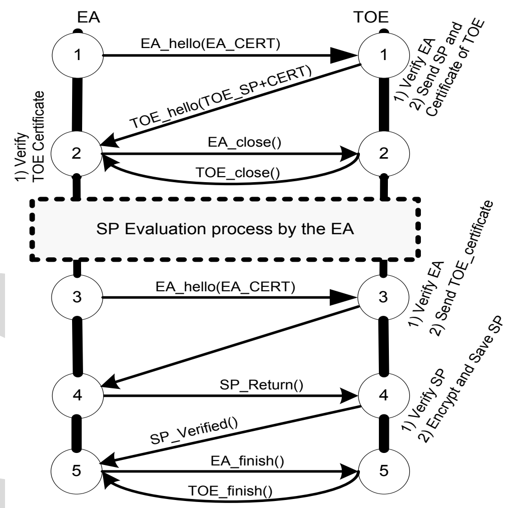

Fig. 1. Node identification and SP exchange between EA and TEO.

For the authentication process, the major components of the protocol are; an evaluation agent (EA), a security profile (SP) representing the system under evaluation (TOE), and an SP-service that provides back-end services and hosts a security profile or ideally a set of profiles for a given group of TOEs. We use the Common Criteria [11] terminology to describe the target of evaluation (TOE). We use the term evaluation agent (EA) to represent the evaluator, evaluation tool, or a principal. An EA is, indeed, a principal (a software agent or code) that remotely collects and processes evaluation information under the control of a RSEP evaluator. That is, EAs collect and process evaluation information stored in security profiles on the back-end service containers (TOEs). To be specific, an EA is a client program that conducts the remote test and evaluation task utilizing the RSEP protocol, where the EA can also perform attacks. It should be noted that, all communications are tunneled with the RSEP encryption scheme in order to ensure authentication and confidentiality, i.e., secure authentication.

Each asset is associated with an SP, which has the function of maintaining current security picture and required attributes of the associated asset. An SP-service is the entry level process that exchanges security profiles with EAs and stores them securely in a disk file. The SP-service waits for requests from the EA and delivers the required documents (i.e., SPs) to the EA. Upon termination of the evaluation of the SP, the EA will return the SP to the SP-service so that the SP-service will store the returned SP in a disk file, which can be further analyzed by the system administrator to see eventual vulnerabilities or improvements.

SP-services are designed to function either as distributed (decentralized) or centralized, [1]. The distributed model is designed to function as a back-end server process (daemon) on the evaluated node, which also hosts the TOE. Clearly, the latter architecture requires numerous SP-services running on each active node. However, ideally, a centralized SP-service is also easily designed to serve a given set of assets or an entire network. The set of assets can also be partitioned from several local area networks if required.

Accordingly, the EA can be either a web client developed separately for conducting remote security evaluations or it can be a module of an SP-service that is accessed via the URL of the centralized SP-service (e.g., web-service container) or the SP-service node.

The SP-service must support development of EAs that can be deployed on different evaluator devices of various platforms and standards, such as a desktop PC, a server host, a hand held device, or any mobile device.

A. *Secure Authentication*

During the authentication, every EA presents a certificate, which is a part of SP. As detailed in [1], each SP, among others, define and store principle identities, issuer of the certificate, legitimate evaluator ID, and current security profile of a given asset. Thus, each asset is also assigned a limited version of SP congaing evaluator's unique ID, evaluator's public key information, name and digital signature of the issuer, timestamp, serial number, version, last evaluation date, period of validity, and digital signature algorithm (the algorithm used to sign the certificate). Though the X.905 digital certificate can be used as an alternative roadmap to designing SPs and EA certificates, we prefer defining a rather lightweight dedicated certificate for use in RSEP. CCITT X.905 v3 [12] defines a standard certificate format for public key certificates and certification validation.

There are a variety of authentication protocols, which are actually applications of cryptographic protocols, [13], used in different context. As more and more systems and public services are linked through Web-services, portals, and integrated applications, the need for a secure authentication standard that allows global authentication becomes more and more apparent. Applied standards often issue a trusted third party to create and distribute certificates and resolve disputes. For example [14], suggests a mutual authentication protocol of two principals with a trusted server and exchange of a new symmetric key, which uses one-way functions and no encryption. On the other hand, authors in [15] present a distributed contract signing protocol claimed to be an abuse-free. That is no party ever can prove to a third party that the principal is able of determining the issue of the exchange (validate or invalidate the contract). To achieve this goal, a special construction called private contract signature is introduced. Such a private contract signature has the particular property that it is meaningful only for a given trusted third party. Moreover, this protocol is optimistic in the sense that the trusted third party is required only in case of disputes. In [16] a secure authentication protocol using uncertified keys is proposed. It is claimed that the protocol is extended to counter the session key compromise problem and to support repeated authentication, in a more secure and flexible way without losing its optimality. A hybrid authentication protocol for large mobile networks is proposed in [17]. Since RSEP also aims supporting wireless networks, this work is an excellent reference for comparison of RSEP's efficiency and the ability of its secure authentication. The designing criteria of the inter-domain authentication protocols include: the scalability, the communication efficiency and the computational efficiency, and the robustness of security. Authors, in [17], discuss weaknesses of some existing protocols against the session key compromise, and then propose a new and efficient inter-domain authentication protocol. Based on public key, challenge-response and hash chaining, this approach simultaneously achieves good scalability, low communication cost and low computational cost, and resistance to the session key compromise attack. Though we did not analyze session key weakness of RSEP, however, it shows useful scalability, low cost and higher computational efficiency. Based on the standard web-technology, implementation RSEP is easy and can provide good interoperability on wider variety of platforms.

III. COMMUNICATION OF AGENTS AND SP-SERVICES

The function of the RSEP protocol, evaluation process, and data exchange between EAs and the TOEs are explained in [5]. However, for the sake of clarity, we need to present a brief description of the protocol here. Initially, a handshake session is required for the communicating nodes (EA and TOE) to mutually and securely authenticate each other. Thus, the interaction starts with a hello message containing the EA's certificate (EA_CERT) sent to the TOE (SP-service). The initial hello message requests a valid certificate from the TOE. On receipt of EA_ID, the TOE verifies the EA. On successful verification, the TOE sends a hello message containing its own certificate comprised of TOE_ID and its SP, steps 1→1 and 1→2 in Fig. 1. On receipt of a valid certificate from the SP-service, the EA terminates the hello session and begins the evaluation of the received SP, step 2→2. Termination of connection must be synchronized, that is, both parties should mutually execute close functions. When the evaluation process is over, the SP is sent back to its owner, i.e., back to the associated TOE, step 4→4. Before sending the SP, a new handshake session must be established, steps 3→3 and 3→4. The security of the data exchange between the EA and TOE is ensured by the combination of two basic services: authentication and confidentiality. An initial certificate exchange composed in the hello session is required for protection against masquerading and "man in the middle" type attacks. Interactions are securely performed by a dedicated encryption scheme, which is mainly based on the secure exchange protocol applying the elliptic curve cryptography (ECC) presented in [18]. Elliptic curve cryptography has been the focus of much recent attention since it offers the highest security per bit of any known public key cryptosystem. The advantage of smaller key sizes makes ECC particularly attractive for embedded and mobile applications since its implementation requires less memory and processing power. Hand-held evaluation agents of RSEP are designed to implement ECC instead of RSA, which requires higher computational power.

Fig. 2 illustrates the transfer of the certificate and the SP of a given TOE from the SP-service.

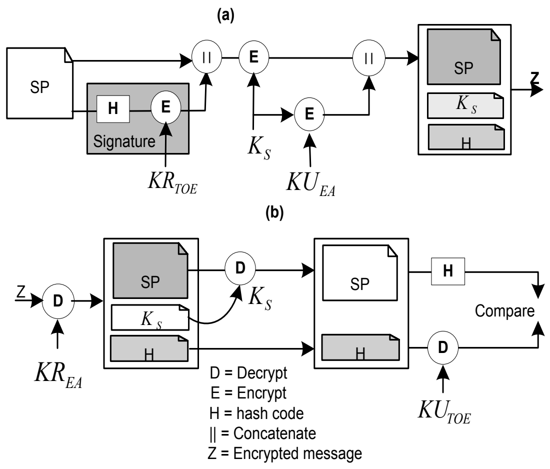

Fig. 1. The secure authentication process.

As shown the security of the data exchange between the TOE and EA is ensured by the combination of two basic services: authentication and confidentiality. Communication is securely authenticated by a dedicated encryption scheme, which is implemented in the RSEP protocol, applying the public key cryptography. It must be noted that the specification of RSEP requires EEC due to the above-mentioned advantages. The interactions assume that the communicating parties, EA and TOEs, have already exchanged required public keys. The exchange scenario shown in Fig. 2 can be expressed as

$$Z = K_S[KR_{TOE}[H(SP)] \parallel SP] \parallel KU_{EA}(K_S),$$

where Z denotes the encrypted data packet, e.g., unique certificate, of the sender. The certificate is composed of the sender’s unique ID, SP and the signature, which further contains actual ID (IP and MAC addresses) of the computer node, and the mandatory fields of the SP specified in [1]. It should be noted that the scenario shown in Fig. 2 is generally applicable, that is, it applies to both directions, i.e., from EAs to TOEs and from TOEs to EAs. In Fig. 2(a) the transfer is from the evaluated node (TOE) to the evaluator (EA), and in Fig. 2(b) the evaluator receives the encrypted data and authenticates the evaluated node (TOE).

The same algorithm and data exchange function are used when the EA is authenticated. In this case the SP will contain the EA related data, KR_{TOE} will be replaced with KR_{EA}, and KU_{EA} will be replaced with KU_{TOE}, and so on.

More precisely, Fig. 2 shows a modified public-key scheme that is used for the SP exchange of the RSEP protocol. The modification is actually achieved by encrypting both the SP and its signature with a Blowfish [19] secret-key using a temporary session key (Ks). The session key generated using the BBS-algorithm [20] is also encrypted with the public key of the receiver, i.e., EA (KU_{EA}), and then concatenated with the encrypted SP and its signature. As shown in Fig. 2(a), the SP’s hash code is generated using SHA-1 [21] and encrypted with the senders (TOE’s) private key KR_{TOE} to produce the digital signature, [22]. The digital signature scheme of RSEP is relatively simple compared to existing schemes. Details on various schemes can be found in [23], [24], and [25]. As illustrated in Fig. 2(b) the receiver, EA, uses its own private key, KR_{EA}, to decrypt the message and obtain both the SP and the session key. The EA uses the decrypted session key to decrypt the SP. Then it uses the sender's public key, KU_{TOE}, to decrypt and recover the hash code. The EA generates a new hash code from the decrypted SP just received, and compares it with the decrypted hash code. If these two codes match then the SP is considered authentic, however, if no match is found, then the receiver replies with an error code, and terminates the authentication session. Obviously; during the secure authentication confidentiality of communication is also ensured. Further details on hashing, various secure authentication schemes, verifiable encryption of digital signatures, and their applications can be found in [26]-[30].

A. The Pseudo-code of the Design

In a modular approach, Java classes of varying code size are grouped into related modules. The following list shows the class prototypes implemented for the scenario shown in Fig. 2. The implementation of the code list given below is used by TOEs, however, replacing the key-pair of TOE-nodes with the key-pair of EA-nodes the implementation can be used by EAs without further modifications. To avoid over complexity we have presented only pseudo-code of the design in APPENDIX.

```
1. Cipher SP ← Encrypt_SP (Plain SP, SecretKey Skey)
2. Plain SP ← Decrypt_SP (Cipher SP, SecretKey Skey)
3. vector<> SP, KSc, Hash ← Parse ( packet Z )
4. Cipher DS ← Enc_Hash_SP_withKR_TOE(Digest
   Hash_SP, PrivateKey KR_TOE)
5. Digest Hash_SP ← Dec_Hash_SP_withKU_TOE(Cipher
   Hash_SP, PublicKey KU_TOE)
6. Cipher KSc ← Enc_KS_withKU_EA(SecretKey KS,
   PublicKey KU_EA)
7. SecretKey KS ← Dec_KSc_withKR_EA(String KSc,
   PrivateKey KR_EA)
8. Enc_DS_withKR_TOE(String DSc, PublicKey KR_TOE)
9. Dec_DSc_withKU_TOE(String DSc, PublicKey KU_TOE)
```

```
10. Digest Hash ← GetHashCode (Plain SP )
11. boolean r ← Verify (Digest HRecv, Digest HGen)
12. Packet Z ← BuildPacket()
13. SaveToDisk (vector M_or_Z)
14. Ciper DS ← SignSP (vector SPp, KRtoe)
15. SecretKey KS ← GetSessionKey()
16. Cipher SPandDSc ← Encrypt_SPandDS (Plain SPandDS,
    SecretKey KS)
17. Plain SPandDSc ← Decrypt_SPandDS (Cipher SPandDS,
    SecretKey KS)
18. String x_pls_y ← Conc(String x, String y, String
    sep)
```

Since the Java cryptography library provides necessary routines, designing the modules is simple. In APPENDIX, only the pseudo-code of the entire design is presented. Detailed implementation of modules can be simplified via calls to the required library routines. For example, to build the hash function we need to import (include) the library `java.security.MessageDigest` in order to make the following code work:

```
public final class GetHashCode
{
  public static GetHashCode instance;
  public GetHashCode(){ }
    public static String get_Hash_SP(String SP){
    MessageDigest md = null;
    try{
       md = MessageDigest.getInstance("SHA");
    }
    catch(NoSuchAlgorithmException e){
      System.err.println("No Algorithm Found");
    }
    try{
       md.update(SP.getBytes("UTF-8"));
    }
     catch(UnsupportedEncodingException e){
       System.err.println("Error while encoding");
     }
     byte HashSP[] = md.digest();
     String Hash_SP =
        new BASE64Encoder().encode(HashSP);
      return Hash_SP;
}
```

As shown in the pseudo-code, the function `get_Hash_SP()` of class `GetHashCode` produces SHA-based message digest of a given SP (or a certificate) and returns the result to the calling object.

IV. CONCLUSION

We have implemented RSEP using the built in encryption library of the Java language. The Java development environment provides a remarkably short development time for an efficient implementation. Due to Java's interoperability and widespread availability, implementation of RSEP in Java has the obvious benefit of use on different platforms. Especially, the implementation of RSEP is a useful way of teaching security protocols and security assessment at universities with network security curriculum. The specifications of the RSEP protocol aimed at providing a secure and efficient remote IT security test and evaluation scheme. We believe that the specifications of the RSEP will inspire implementors and security product developers with rapid and standard-friendly implementations of high quality tools and techniques for remote test and evaluations of a multitude discipline of systems. The RSEP scheme improves security awareness efficiently and introduces means for early detection and tracking of hazards, threats, and vulnerabilities. It has also the ability to ensure implementation of proactive lifecycle security policies by acting before the incident occurs. The RSEP scheme specifies the following features:

- It provides a model for the achievement of proactive lifecycle security (PLS). For the first step in achieving the PLS, this scheme forces TOEs to continuously track threats and vulnerabilities and recover from incidents.
- It supports simplified, time- and location-neutral, and secure evaluations over the Internet and other open networks.
- It provides models and specifications of operations for inexpensive implementation of evaluation tools and techniques.
- It provides very efficient time usage by reducing the overall test and evaluation process for an entire network of any size, thus, as a result of achieving PLS, the timeliness requirement is fulfilled.
- Monitoring the security of the security systems: RSEP provides models and means to remotely monitor protection mechanisms in a timely manner. In this manner dynamic threat patterns can be captured and analyzed in order to predict future security risks encountered by the protection mechanisms.

RSEP uses an informal method to compute quantitative risk values. Quantitative risk assessment methods are relatively difficult to realize than that of the qualitative methods. This problem can likely be solved if a formalized quantitative risk assessment method could be defined. By applying the specifications of the formalized quantitative risk assessment method with an interoperable interface, any non-RSEP compliant risk assessment method can be used to obtain ideal results for quantitative risk values.

APPENDIX

Module 1:

```
Encrypt_SP(Plain SP_buffer, SecretKey Skey){
 int i = 0;
   while ! eof ← SP_buffer do
     line = SP_buffer[i];
      int j = 0;
  while ! eof ←line do
    savetofile (encrypt(line[j], Skey));
    /* Decryption is similar to encryption, i.e.,
            1)  Extract cipher SP from file
            2)  Parse the SP into characters as above
            3)  Decrypt_SPc(line[j], Skey));
         */
    j++;
  }endwhile
  i++;
   } endwhile
}

Decrypt_SPc (Cipher SPc, SecretKey SK)
{
  return (D(SPc, SK);
}

SignSP (Plain SPp, PrivateKey KR_TOE)
{
  GetHashCode HashSP;
        Digest hash = HashSP.get_Hash_SP (SPp)
        DS = Enc_Hash_SP_withKR_TOE(hash, KR_TOE)
        return DS;
}

SaveToDisk(vector SP, vector H_P,vector H)
{
        if ( Verify (H_P,H) )
         write_to_disk( SP );
        else
         save_error();
}

String x_pls_y ← Conc(String x, String y, String
Sep)
{
        return x + Sep + b;
}
```

Module 2:

```
Parse (Packet Z)
{
  vector < string >  KS,SPc,Hc;
  p ←z;
  while (*p)! = '#' do
  KS.push (*p); ++p;
  Endwhile
  ++p;
  while (*p)! = '#' do
      SPc.push (*p); ++p;
  Endwhile
  ++p;
  while (*p)! = EOF do
    Hc.push(*p);
    ++p;
  endwhile
  return vector < > KS,SPc,Hc;
}
```

Module 3:

```
Dec_KSc_withKR_EA(String KSc, PrivateKey KR_EA)
{
       return (D(KSc, KR_EA );
}

Enc_KS_withKU_EA(SecretKey KS, PrivateKey KU_EA)
{
       return (E(KS, KU_EA );
}
```

Module 4:

```
GetHashCode ( vector SP )
{
    return (SHA-1(SP));
}

Enc_Hash_SP_withKR_TOE(Digest Hash_SP, PrivateKey
KR_TOE)
{
  return (E(Hash_SP, KR_TOE));
}

Dec_Hash_SP_withKU_TOE(Cipher Hash_SP, PublicKey
KU_TOE)
{
    return (D( Hash_SP, KU_TOE );
}
```

Module 5:

```
Encrypt_SPandDS (Plain SPandDS, SecretKey KS)
{
           return(E(SPandDS, KS);
}

Decrypt_SPandDS (Cipher SPandDS, SecretKey KS)
{
           return(D(SPandDS, KS);
}
```

Module 6:

```
Verify (Digest HashRecv, Digest newHash )
{
    if( compare(HashRecv, newHash) ) {
     return true;
    }
    return false;
  }
```

Module 7:

```
MakePacket SP()
{
    Get SP from file,
    Get public key pair (KU_EA and KR_TOE);
    Plain SP = Decrypt_SPc (Cipher SP, SecretKey
FKS);
    Digest Hash = Hash_SP (Plain SP);
    Signature DS = Enc_Hash_SP_withKR_TOE(Digest
    Hash, PrivateKey KR_TOE);
    Plain SPandDS = conc (Plain SP, Signature DS);
    Generate session key KS;
    SecretKey KS = GenSessionKey();
    Cipher KS_C = Enc_KS_withKU_EA(SecretKey KS,
    PublicKey KU_EA);
    Cipher SPandDS_C = Encrypt_SpandDS (Plain SPandDS,
    SecretKey KS);
    Make_Z = Conc ( Cipher SPandDS_C, Cipher KS_C );
}
```

Module 8:

```
GetSessionKey( )
{
do{
        ( testRandom( K ) != true )
        // Generate Blum Blum Shub number
         HugeInt K = BBS( );
}
while(!randomness);// Test for beter randomness
    return K;
}
```

Module 9:

```
Enc_DS_withKR_TOE(String DS, PublicKey KR_TOE)
{
  return(E(DSc, KR_TOE);
}
Dec_DSc_withKU_TOE(String DSc, PublicKey KU_TOE)
{
  return(D(DSc, KU_TOE);
}
```

ACKNOWLEDGMENT

I would like to thank my graduate students at the Faculty of Computer Sciences at Izmir University of Economics for their support in coding the RSEP protocol: Derya Bıyık, Gizem Sezen, Kahraman Akyıl, M. Egemen Özkan, Murat Yatağan, M. Mesut Özışık, and Melih Yenel.

REFERENCES

[1] S. Kondakci, "A Remote IT security evaluation scheme: A proactive approach to risk management", in *Proc. 4th IEEE -IWIA*, 2006, pp. 93-100.

[2] B. Schneier, *Applied Cryptography*, New York: Wiley, 1996.

[3] Woo, T. Y. and Lam, S. S. 1994. "A lesson on authentication protocol design", *SIGOPS Oper. Syst. Rev.*, 28, 3, 1994, pp. 24-37.

[4] A. Liebl, "Authentication in distributed systems: a bibliography", *SIGOPS Oper. Syst. Rev*., 27, 4, 1993, pp. 31-41.

[5] S. Kondakci, "Remote security evaluation agent for the RSEP protocol", in *Proc. International Conference on Security of Information and Networks (SIN 2007),* Trafford Publishing, ISBN:978-1-4251-4109-7, 2007, pp. 186-195.

[6] A. Weimerskirch, G. Thonet, "A Distributed Light-Weight Authentication Model for Ad-hoc Networks", in *Proc. 4th int. Conf. Seoul on information Security and Cryptology, Lecture Notes In Computer Science*, vol. 2288, Springer-Verlag, 2001, pp. 341-354.

[7] M. Bohge, W. Trappe, "An authentication framework for hierarchical ad hoc sensor networks", in *Proc. 2nd ACM Workshop on Wireless Security,* WiSe '03, 2003, pp. 79-87.

[8] S. C. Huang, H. Zhu, and W. Zhang, "SAP: seamless authentication protocol for vertical handoff in heterogeneous wireless networks", in *Proc. 3rd international Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, vol. 191, 2006, 32.

[9] CSI, *2006 CSI/FBI Computer Crime and Security Survey*, CSI Computer Security Inst. Publications, 2007.

[10] D. Gardon, "IT's confidence crisis", *2006 InfoWorld Security Survey*, Oct. 2006.

[11] *Common Criteria/ISO IS 15408, Version 2.1*, October 1999, http://csrc.nist.gov/cc/ccv20/ccv2list.htm.

[12] *RFC 2459*, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", http://www.ietf.org/rfc/rfc2459.txt, accessed 2006.

[13] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols", *IEEE Transactions on Software Engineering*, 22(1), 1996, pp. 6-15.

[14] Li Gong, "Using one-way functions for authentication", *Computer Communication Review*, 19(5), 1989, pp. 8-11.

[15] J. A. Garay, M. Jakobsson, and P. MacKenzie, "Abuse-free optimistic contract signing", in *Proc. Crypto'99*, *Lecture Notes in Computer Science*, Springer-Verlag, vol. 1666, 1999, pp. 449-466.

[16] I. Kao and R. Chow, "An efficient and secure authentication protocol using uncertified keys", *SIGOPS Oper. Syst. Rev*, 29, 3,1995, pp. 14-21.

[17] H. Chien and J. Jan, "A hybrid authentication protocol for large mobile network", *J. Syst. Softw*. 67, 2, 2003, pp. 123-130.

[18] A. Enge, *Elliptic Curves and Their Applications to Cryptography— An Introduction*, Kluwer Academic Publishers, 1999

[19] B. Schneir, "Fast software encryption", *Cambridge Security Workshop Proceedings*, Springer-Verlag, 1994, pp. 191-204.

[20] L. Blum, M. Blum, M., and M. Shub, "A simple unpredictable pseudo-random number generator, *SIAM Jnl on Computing*, No. 2, 1986.

[21] *FIPS 180-1, Secure Hash Standard, Secure Hash Algorithm,* http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf.

[22] R. Rivest, A. Shamir, and L. Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, v.21 n.2, 1978, p.120-126.

[23] A. Shamir, "Identity Based Cryptosystems and Signature Schemes", in *Proc. Advances in Cryptology Crypto' 84*, *Lecture Notes in Computer Science*, volume 0196, Springer-Verlag, 1984.

[24] R. Jueneman, S. Matyas, and C. Meyer, "Message authentication*", IEEE Communications Magazine*, Vol. 23, No. 9, 1985, pp. 29-40.

[25] F. Hess, "Efficient identity based signature schemes based on pairings". in *Proc. of SAC'02, Lecture Notes in Computer Science*, Springer-Verlag, 2002.

[26] G. Ateniese, "Verifiable encryption of digital signatures and applications", *ACM Trans. Inf. Syst. Secur*. 7, 1 2004, pp. 1-20.

[27] S. Čapkun and M. Čagalj, "Integrity regions: authentication through presence in wireless networks", in *Proc. 5th ACM Workshop on Wireless Security,* WiSe '06, 2006, pp. 1-10.

[28] R. Safavi-Naini, S. Wang, and Y. Desmedt, "Unconditionally secure ring authentication", in *Proc. 2nd ACM Symposium on information, Computer and Communications Security,* 2007, pp. 173-181.

[29] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", 1997.

[30] N. Aboudagga, M. T. Refaei, and M. Eltoweissy, L. A. DaSilva, and J. Quisquater, "Authentication protocols for ad hoc networks: taxonomy and research issues, in *Proc. 1st ACM international Workshop on Quality of Service &Amp; Security in Wireless and Mobile Networks.* Q2SWinet '05, 2005, pp. 96-104.