Journal of Theoretical and Applied Information Technology
15th November 2020. Vol.98. No 21

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A CRYPTOGRAPHIC KEY MANAGEMENT SYSTEM MODEL

[1]SAULE NYSSANBAYEVA, [2]ARMANBEK HAUMEN, [3]ANDREY VARENNIKOV, [4]NURSULU KAPALOVA

[1,2,3,4]Institute of Information and Computational Technologies

CS MES RK Almaty, Kazakhstan

E-mail: [1]snyssanbayeva@gmail.com, [2]haumen.armanbek@gmail.com, [3]avarennikov@gmail.com, [4]kapalova@ipic.kz

ABSTRACT

This paper describes the requirements and issues that need to be considered and taken into account when developing cryptographic key management systems. We developed the functional structure of a cryptographic key management system, as well as a cryptographic protection scheme for messages using encryption and electronic digital signature based on the OpenPGP protocol. We propose modifications of the ElGamal scheme based on non-positional polynomial notations to use as encryption and electronic digital signature algorithms. Also, an assessment of the enumeration of all possible options for the selection of working bases and secret keys is given.

Keywords: *Cryptographic, Cryptographic Keys, Key Management, Encryption Algorithm, Asymmetric Cryptosystems.*

INTRODUCTION

One of the main means used to protect the information in computer systems is cryptographic transformations. Modern cryptographic methods use cryptographic keys and a cryptographic key management system. Key management is the most critical area for cryptographic applications. It is easy to take advantage of cryptographic technologies, while it is much more difficult to store, use, and exchange keys [1].

The issue of key updatingis directly related to one of the key management problems – key distribution. There are several approaches to distribute keys[2]:

- Physical distribution.
- The issuance of a shared key to the participants of the interaction by a key distribution center – a "subscriber encryption" scheme.
- A certification authority provides keys for access to public keys of users and issues secret keys to users.
- A network of trust. The network is used in asymmetric cryptosystems. Users themselves distribute their keys and monitor the keys of other users.
- Key exchange protocols. A secret key is generated and exchanged through unprotected communication channels between interaction participants who had not previously shared any secret key.

The disadvantage of the methods using a key distribution center is that the center knows who and which keys are assigned, therefore, the center may read all messages circulating in the information system. With direct key exchange, a problem of authentication of subjects arises.

The paper discusses the general principles of designing cryptographic key management systems, the functional structure of a cryptographic key management system, and methods for distributing cryptographic keys using self-certified keys. We propose a scheme for the open distribution of cryptographic keys based on a modified ElGamal encryption system and electronic digital signature**.**

1. GENERAL PRINCIPLES TO DESIGN CRYPTOGRAPHIC KEY MANAGEMENT SYSTEMS

Cryptography is most often used to protect information from unauthorized disclosure, detect unauthorized modifications, and confirm the authenticity of a data source. Cryptographic methods use the cryptographic keys that are managed and protected by a Cryptographic Key

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Management System (CKMS). The following is a standard list of CKMS functions [3-6]. There is no need for a cryptographic key management system to implement all of the above functions, since some functions may not be applicable.

1) Cryptographic key generation.

2) Registration of key owners (individuals, organizations, devices, or processes).

3) Key activation. This function provides the transfer of a cryptographic key from the standby state into the active state.

4) Key deactivation. This function terminates a key activity by transferring this key into the corresponding state.

5) Key cancellation. This function is used in cases where the use of a key must be stopped before the end of the cryptoperiod set for this key.

6) Key suspension and reactivation.

7) Extension of the cryptoperiod for public keys.

8) Key generation based on a shared secret. This function is often used in key-agreement protocols to obtain a common key.

9) Destruction of keys and their metadata.

10) Associating keys with metadata. Depending on the nature of the information stored in a metadata element, the metadata element may require confidentiality protection, integrity protection, and source authentication.

11) Changing metadata associated with a key.

12) Deleting metadata associated with a key. Metadata elements can be deleted all at once, and also as separate elements or as a given subset of elements.

13) Obtaining a list of key metadata. This function allows an object to obtain a list of metadata elements of a given key, provided that the object has the right to do so.

14) Storage of operational keys and their metadata.

15) Backing up keys and metadata. Backups of keys and metadata can be stored in the same place as the operational keys/metadata, or in another place to ensure that these keys and metadata could be restored if necessary.

16) Archiving keys and their metadata – includes storing keys and metadata in a secure, durable storage medium so that they could be restored if necessary.

17) Recovery of keys and their metadata – includes obtaining a copy of the keys and their metadata that were previously stored in online storage, backup, or archive.

18) Secure key distribution between two or more objects. A key can be transferred from one object to another (key transfer) or can be obtained from the information provided by each of these objects (key agreement).

19) Loading keys and their metadata into a cryptographic module (a set of hardware, software, or firmware that implements the necessary information security functions).

20) Unloading keys and their metadata from a cryptographic module.

21) Validation of parameters used by some cryptographic algorithms.

22) Validation of a public key to verify its arithmetical correctness. These verifications usually depend on the algorithm for which the given public key is intended, but do not depend on the knowledge of the corresponding private key.

23) Verification of a public key certification path (also known as a certificate chain).

24) Validation of symmetric keys and their metadata. For example, these checks may include the verification of integrity, length, and format of a key.

25) Validation of a private key to ensure that it complies with the prescribed specifications. This verification can only be performed by the owner of the private key or by a trusted third party acting on behalf of the owner of the private key. This verification may also include the consistency checking of the key pair to make sure that the function applied to the private key is complementary to the function applied to the public key.

26) Verification of ownership of a private key. This function is used by an object that receives a public key and wants to make sure that the claimed owner of the public key does indeed have the corresponding private key and, therefore, is the owner of the key pair.

27) Performing cryptographic functions using a key. These functions can include digital signature generation, digital signature verification, data encryption, data decryption, key encryption, key decryption, generation and verification of message authentication codes used to detect both accidental and intentional data changes.

Managing trusted public key certificates used to establish trust in other public keys. Trust in such public keys is established by verifying all signatures in the chain of public-key certificates, starting with the key that the verifying object trusts.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. FUNCTIONAL STRUCTURE OF A CRYPTOGRAPHIC KEY MANAGEMENT SYSTEM

Figure 1 shows the logic diagram of the key management process.

The calling entity makes a call to one of the functions that are handled by the cryptographic key management system. The CKMS uses a secure cryptographic key store and interacts with the cryptographic module designed in the form of software, hardware, or firmware and implementing cryptographic functions. If the authentication of the calling entity was successful and the entity is properly authorized, then the called function is performed by sending queries to the cryptographic module. Finally, the response received from the cryptographic module is passed back to the caller.

The cryptographic key store is a normalized relational database. The purpose of normalization is to prevent duplication of stored data, as well as the corruption of consistency that may occur when updating data. The database structure is normalized by dividing the data into separate information entities, which leads to the creation of several interconnected tables.

The logical structure of the cryptographic key store is shown in Figure 2.

The cryptographic key store contains the following information entities:

1) Cryptographic keys (symmetric and asymmetric), as well as their metadata, including a unique identifier for a key, a key name, a pointer to a key type, the start date and end date of the key cryptoperiod, the current state of a key, etc.). Secret keys (symmetric and asymmetric private keys) are stored in an encrypted form.

2) Types of cryptographic keys (e.g., digital signature keys, data encryption keys, key-encryption keys, etc.).

3) Key parameters, each of which consists of a type and value (for example, working bases used by encryption and digital signature algorithms based on a non-positional polynomial notation (NPN)).

4) Asymmetric key pairs, each of which consists of a public and corresponding private key. Key pairs form a set of own keys. Non-pair public keys form a set of public keys.

5) Possible states of cryptographic keys (for example, "Pending putting into action", "Active", "Put out of action", etc.).

6) Certificates confirming the validity of public keys and establishing the key life (which may be shorter than the cryptoperiod).

7) Certificate types (PGP or X.509).

8) A list of revoked public key certificates containing the date and reason for revocation.

9) Registered users of the cryptographic key management system, including their unique identifier and password for entering the system (in encrypted form).

10) Types of registered users (e.g. individuals, cryptographic modules, software applications, etc.)

11) Levels of trust in system users, expressed in numerical values.

12) A control list of system users having access to cryptographic keys (including specifying key owners).

13) An audit log containing information about events that are related to cryptographic keys, indicating the exact date and time of these events. Such events include generating keys, changing the state of keys during their life cycle, revoking keys, destroying keys, etc.

3. SECURE MESSAGE EXCHANGE SCHEME

The secure message exchange is based on the OpenPGP protocol [7], which combines symmetric key encryption and public key encryption for confidentiality. To achieve confidentiality, a message is encrypted using a symmetric encryption algorithm. Each symmetric key (called a session key) is used only once, and only for one message. Since the symmetric key is used only once, it is bound directly to the message and transmitted along with it. To protect the session key, it is encrypted with the recipient's public key. A digital signature using public keys is used to ensure the authenticity of the sender and the integrity of the message.

Figure 3 shows the scheme of secure message transmission using encryption and electronic digital signature.

The sender (sender A) takes the following steps:

1) Decryption of the private key of sender A:

- CKMS extracts the encrypted private key PR_A of sender A from the private key set using the identifier ID_A of sender A as an index.
- CKMS asks sender A for the passphrase P_A to decrypt its private key PR_A.
- CKMS generates a digest of the sender A's passphrase P_A using the hashing algorithm H.
- CKMS decrypts the private key PR_A of sender A with the symmetric decryption algorithm DC using the passphrase P_A of sender A as the digest key.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2) Digital signature generation for plaintext message M:

- CKMS generates a digest of the plaintext message M using the hashing algorithm H.

- CKMS generates a digital signature for the plaintext message M by encrypting the digest of the plaintext message M using the asymmetric encryption algorithm EP and the private key PR_A of sender A.

- CKMS combines the identifier $KI_A$of the public key PU_A of sender A, corresponding to the private key PR_A of sender A, the digital signature of the plaintext message M and the plaintext message M using the concatenation operation and generates a message component $C_1 = KI_A \parallel EP[PR_A, H(M)] \parallel M$.

3) Encryption of the message component C_1:

- CKMS compresses the message component C_1 with the compression algorithm Z.

- CKMS generates a symmetric session key K_s using the pseudo-random number generation algorithm RNG.

- CKMS encrypts the compressed message component C_1 with the symmetric encryption algorithm EC using the session key K_s and produces the encrypted message component $C_2 = EC[K_s, Z(C_1)]$.

4) Generation of a digital signature of the session key K_s:

- CKMS generates a digest of the session key K_s using the hashing algorithm H.

- CKMS produces a digital signature of the session key K_s by encrypting the digest of the session key K_s using the asymmetric encryption algorithm EP and the private key PR_A of sender A.

- CKMS combines the identifier KI_A of the public key PU_A of sender A, corresponding to the private key PR_A of sender A, the digital signature of the session key K_s and the session key K_s using the concatenation operation and generates the session key component $C_3 = KI_A \parallel EP[PR_A, H(K_s)] \parallel K_s$.

5) Encryption of the C3 session key component:

- CKMS extracts the public key PU_B of receiver B from the public key set using the identifier $ID_B$of receiver B as an index.

- CKMS encrypts the component C_3 of the signed session key with the asymmetric encryption algorithm EP using the public key PU_B of receiver B and generates the encrypted session key component$C_4 = EP[PU_B, C_3]$.

6) Generation of the encrypted message protected by the digital signature.

- CKMS combines the identifier KI_B of the public key PU_B of the receiver B, the encrypted session key component C_4 and the encrypted plaintext message component C_2 using the concatenation operation and generates the encrypted message $R = KI_B \parallel C_4 \parallel C_2$, which can be saved on the hard disk or sent to receiver B.

$R = KI_B \parallel EP[PU_B, KI_A \parallel EP[PR_A, H(K_s)] \parallel K_s] \parallel EC[K_s, Z(KI_A \parallel EP[PR_A, H(M)] \parallel M)]$

Figure 4 shows the scheme of processing an incoming message using algorithms for decryption and verification of the electronic digital signature.

The receiving party (receiver B) follows these steps:

1) Decryption of the private key PR_B of receiver B:

- CKMS extracts the encrypted private key PR_B of receiver B from the own key set using the identifier ID_B of receiver B as an index.

- CKMS asks the sender B for the passphrase P_B to decrypt the private key PR_B.

- CKMS generates a digest of passphrase P_B using the hashing algorithm H.

- CKMS decrypts the private key PR_B of receiver B with the symmetric decryption algorithm DC using the passphrase P_B of receiver B as the digest key.

2) Decryption of the digital signature of the session key K_s.

- CKMS decrypts the session key component of the encrypted message R with the asymmetric decryption algorithm DP using the private key PR_B of receiver B.

- CKMS extracts the public key PU_A of receiver A from the public key set using the identifier KI_A of receiver A from the session key component as an index.

- CKMS decrypts the session key digest K_s from the session key component with the asymmetric decryption algorithm DP using the public key PU_A of sender A.

3) Verification of the digital signature of the session key K_s.

- CKMS generates a digest of the session key K_s using the hashing algorithm H.

- CKMS performs a bitwise comparison of the session key digest K_s from the session key component and the session key digest Ks generated in the previous step. Differences between them indicate data integrity damage and further processing is terminated.

4) Decryption of the digital signature of the plaintext message M.

- CKMS decrypts the compressed component of the message M with the symmetric decryption algorithm DC using the session key Ks.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- CKMS decompresses the component of the message *M* using the algorithm Z^{-1}inverse to the compression algorithm Z.

- CKMS extracts the public key PU_A of receiver A from the public key set using the key identifier KI_A of receiver A from the message component as an index.

- CKMS decrypts the digest of the plaintext message *M* from the message component with the asymmetric decryption algorithm DP using the public key PU_A of the sender A.

5) Verification of the digital signature of the plaintext message *M*.

- CKMS generates a digest of the plaintext message *M* using the hashing algorithm H.

- CKMS performs a bitwise comparison of the digest of the plaintext message *M* from the plaintext message component and the digest of the plaintext message *M* generated in the previous step. Differences between them indicate data integrity damage and further processing is terminated.

4. OPEN DISTRIBUTION OF CRYPTOGRAPHIC KEYS USING SELF-CERTIFIED KEYS

Known public key schemes were developed after the concept of public-key cryptography proposed in 1976 [8]. In such schemes, each user has a pair of keys (SK, PK). The first, SK, is a secret key known only to the user. The second, PK, is a public key. These secret and public keys are mathematically related.

Asymmetric encryption systems, due to the complex calculations required by the algorithms, are inferior in computing speed to symmetric encryption systems.

Symmetric cryptosystems are fast, but they have one significant drawback, namely the need for secure key transfer. Asymmetric crypto systems serve as a tool to overcome this drawback, For example, for a symmetric algorithm, a random session key is generated. Such a key, as a rule, has a size from 128 to 512 bits (depending on the algorithm). This key is used in symmetric algorithms to encrypt a message. The random key itself must be encrypted using the public key of the message recipient, and at this stage, a public key cryptosystem is used. Since the session key is short, it does not take long to encrypt it. Then it suffices to send a message encrypted by a symmetric algorithm, as well as the corresponding key in encrypted form. The receiver first decrypts the key with his private key, and then, with the received key, he obtains the whole message.

The key management system under development is intended for symmetric encryption algorithms (developed as part of other projects of the information security laboratory of the RK MES CS IICT). In this system, each symmetric key (called a session key) is used only once, and only for one message. To protect the session key, it is encrypted with the recipient's public key.

To ensure the authenticity (authentication) of a sender and the integrity of a message, a variety of methods and approaches are used, including self-certifying public keys.

The CCITT X509 recommendation uses a digital signature [9], and in the schemes introduced by Shamir, the public key is nothing more than an identifier [10-12], in [13-15] the authors use the RSA digital signature scheme and public keys with self-certification.

The proposed scheme is based on factorization and discrete logarithm problems. Since the public key is at the same time a guarantee of its authenticity, it can be called self-certified, and each user has three attributes (I, SK and PK). Self-certified public-key schemes are based on neither certificates, since there is no separate certificate, nor identification data, since the public key is not limited to the identification. As a result, they help reduce storage and computation (in particular, they do not require hash functions at the authority level), while private keys are still selected by the user and remain unknown to other participants (including the trusted center).

The developed scheme is based on a modified ElGamal encryption system and the electronic digital signature.

5. A MODIFIED ASYMMETRIC ENCRYPTION SYSTEM AS PER THE ELGAMAL SCHEME

Based on the results of the study of the distribution schemes of cryptographic keys and approaches to ensure the authentication of keys, we developed an open cryptographic key distribution scheme using a previously proposed modified asymmetric ElGamal encryption system. The results of the development and study of the modified asymmetric encryption system as per the ElGamal scheme using non-positional polynomial notations (NPNs) were published in [15-16] (Fig. 3). The proposed scheme is based on a hybrid of the modified ElGamal system and the RSA scheme.

The previously developed non-traditional asymmetric encryption algorithm for electronic

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

message M as per the El-Gamal scheme is as follows [14].

1) First of all, an NPN is formed with its working bases consisting of selected irreducible polynomials

$$p_1(x), p_2(x), \dots, p_S(x) \quad (1)$$

over $GF(2)$ of degrees m_1, $m_2, \dots, m_S$ respectively. The polynomials (1) subject to their arrangement constitute a certain base system. All bases are to be different including the case when they have the same degree (to comply with the Chinese remainder theorem). The working range of the NPN is specified by the polynomial (modulus)

$$P_S(x) = p_1(x)p_2(x) \cdots p_S(x)$$

of degree $m = \sum_{i=1}^{S} m_i$.

In NPNs, any polynomial $F(x)$ of degree less than m has a unique representation in the form of

$$F(x) = (z_1(x), z_2(x), \dots, z_S(x)), \quad (2)$$

where $F(x) \equiv z_i(x) (\mathrm{mod}\, p_i(x)), i = \overline{1,S}.$ The positional representation of $F(x)$ is regained from its nonpositional form (2) as follows:

$$F(x) = \sum_{i=1}^{S} z_i(x) B_i(x)$$

, where

$$B_i(x) = \frac{P_S(x)}{p_i(x)} M_i(x) \equiv 1 (\mathrm{mod}\ p_i(x)). \quad (3)$$

The polynomials $M_i(x)$ are selected in such a way as to satisfy the congruence in (3).

2) For each base $p_i(x)$, a primitive element (polynomial) $\alpha_i(x)$ is selected from the complete residue system modulo $p_i(x)$, i.e. the degrees of $\alpha_i(x)$ are not higher than m_i, where $i = \overline{1,S}$. Then the primitive element in the modified unconventional encryption algorithm is interpreted as a sequence of residues from dividing some polynomial $\alpha(x)$ into the bases $p_1(x), p_2(x), \dots, p_S(x)$ respectively:

$$\alpha(x) = (\alpha_1(x), \alpha_2(x), \dots, \alpha_S(x)),$$

where $\alpha(x) \equiv \alpha_i(x) (\mathrm{mod}\ p_i(x))$, $i = \overline{1,S}$.

The selected working bases and their corresponding primitive polynomials $\alpha_i(x)$ are kept secret.

To restore the result in a positional form, its residues are used to determine the basis of the NPN by the formula (3). For this, the polynomials $\delta_i(x) \equiv \frac{P_S(x)}{p_i(x)} (\mathrm{mod}\ p_i(x))$ and their inverses $\delta_i^{-1}(x) \cdot \delta_i(x) \equiv 1 (\mathrm{mod}\ p_i(x))$ are calculated. Then the bases, which are also secret parameters of the algorithm, are found by the formula $B_i(x) = \delta_i^{-1}(x) \cdot \frac{P_S(x)}{p_i(x)}$.

3) Next users A and B, independently from each other, select personal (private) keys l_A and l_B, respectively, so that $1 < l_A, l_B < 2^m$.

4) Then users A and B calculate the third element of the public key, respectively:

$$\beta_A(x) = (\beta_{A_1}(x), \beta_{A_2}(x), \dots, \beta_{A_S}(x))$$

,

where $\beta_{Ai}(x) \equiv \alpha_i^{l_A}(x) (\mathrm{mod}(p_i(x)),\ i = \overline{1,S}$;

$$\beta_B(x) = (\beta_{B_1}(x), \beta_{B_2}(x), \dots, \beta_{B_S}(x))$$

,

where $\beta_{Bi}(x) \equiv \alpha_i^{l_B}(x) (\mathrm{mod}(p_i(x)),\ i = \overline{1,S}$.

All exponentiation operations are performed in a non-positional polynomial notation, therefore, the calculation of these operations can be performed in parallel modulo the polynomials selected as the base of the system.

5) After that, parties A and B exchange the calculated values of the public keys $K_A(x) = (P_S(x), \alpha(x), \beta_A(x))$, $K_B(x) = (P_S(x), \alpha(x), \beta_B(x))$, respectively, over an insecure channel in binary representation.

6) Using the public keys of the recipient, users A and B perform the process of encryption E_k of the message M similar to the traditional ElGamal scheme: $E_k(M) = (C_1, C_2)$, where $C_1 = \alpha^r (\mathrm{mod}(P_S(x))$, $C_2 = M \cdot \beta^r (\mathrm{mod}(P_S(x))$, where r is a randomly selected number (randomizer) and $0 \le r \le 2^m$ (also for each working base one can choose different randomizers).

7) To decrypt with D_k an encrypted message, users A and B use their private keys by the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

formula:

$D_k(C_1, C_2) = C_2 \cdot (C_1^{l_i})^{-1} \bmod P_S(x) = M$, where $i = A, B$.

All calculations in the NPN can be performed in parallel modulo the work bases $p_1(x), p_2(x),..., p_S(x)$, which makes it possible to significantly increase the processing speed.

6. A MODIFIED SYSTEM OF ELECTRONIC DIGITAL SIGNATURE AS PER THE ELGAMAL SCHEME

The scheme under consideration assumes that users on the principle of trust can confirm the authenticity of each other's public keys. Suppose users *A* and *C* trust user *B*. To establish trust between *A* and *C*, user *B* acts as a guarantor for key authentication.

As described above, the public keys of users *A*, *B*, and *C* are $K_A(x) = (P_S(x), \alpha(x), \beta_A(x))$, $K_B(x) = (P_S(x), \alpha(x), \beta_B(x))$, and $K_C(x) = (P_S(x), \alpha(x), \beta_C(x))$ respectively. In order for public keys to have the property of self-certification (without a trust center), we use the modified ElGamal EDS system based on NPNs. A brief description is given below.

1) The process of calculating public and private keys (paragraphs 1-5 of Section 2) is carried out in accordance with the encryption algorithm;

User B then signs *C*'s public key and establishes trust between users *A* and *C* to authenticate C's keys to user A.

2) The signature is calculated in the same way as the ElGamal EDS algorithm;

- Using his/her private key, user *B* performs the process of generating a digital signature of the identifier I_C and the public key K_C of user *C*.

- Selects random numbers r_i such that $1 \le r_i < n_i = (2^{m_i} - 1)$ and $GCD(r_i, n_i) = 1$, where m_i is the degree of the polynomial$p_i(x), i = \overline{1,S}$.

- Finds inverse elements $r_i^{-1}, i = \overline{1,S}$.

- Calculates the first part of the digital signature $S_i'(x) = (\alpha_i(x))^{r_i} \bmod p_i(x), i = \overline{1,S}$.

- Calculates the second part of the digital signature $S'' = r_i^{-1}(M_i - l_B \cdot S_i')(\bmod n_i), i = \overline{1,S}$, where l_B is the secret key of user B, and M_i is the *i*th message fragment $M = h(I_c, K_c)$, where h is some hash function. I_C and K_c are respectively the identifier and public key of user *C*.

3) Then the respective public key of *C* signed by *B* is published: $Sig_B(K_C) = (S_i'(x), S_i'') : (K_c, Sig_B(K_C))$.

4) Next, user *C* proves to user *A* that he is exactly *C*. This can be done by verifying the signature of B.

- To verify the signature of user B using the public key of user *C*, user A must do the following:

1. Get the public key of user C $K_C(x)$.
2. Calculate the first part of the check value:

$V_i'(x) = (\beta_i(x))^{S_i'} \cdot (S_i'(x))^{S_i''} \bmod p_i(x), i = \overline{1,S}$.

3. Calculate the second part of the check value:

$V_i''(x) = (\alpha_i(x))^{M_i} \bmod p_i(x),, i = \overline{1,S}$

5) If $V_i' = V_i''$, $i = \overline{1,S}$, then the signature is accepted. That is, user A establishes trust for user *C* since his public key is signed by trusting user *B*.

7. ANALYSIS OF THE STRENGTH OF THE MODIFIED ALGORITHMS BASED ON NON-POSITIONAL POLYNOMIAL NOTATIONS

In this section, the "cryptographic strength" of the modified algorithms is measured as exhaustive secret-key testing (brute force attack). Breaking an NPN system and the selection of primitive elements for them are calculated in the same way.

Working bases are selected subject to the equation:

$$k_1 m_1 + k_2 m_2 + \cdots + k_S m_S = m; \quad (4)$$

where $0 \le k_i \le n_i$ is the number of selected irreducible polynomials of degree m_i, n_i is the number of all irreducible polynomials of degree m_i, $1 \le m_i \le N$, $i=1,2,..,S$, $S = k_1 + +k_2 + ... + k_s$ is the number of selected working bases. Equation (4) determines the number *S* of irreducible polynomials of various degrees that can be chosen as the bases of a system, the residue representation for which covers the length m of the public keys of the modified crypto algorithms.

One solutionof equation (4) is the specific values of the coefficients $k_1, k_2, \ldots, k_S$, which define one system of bases. The number of solutions of equation (4) determines the number of base systems of the NPN, while each system accounts for all possible permutations of the bases $p_1(x), p_2(x), \ldots, p_S(x)$. With an increase in the degree of irreducible polynomials, their number goes up significantly, and accordingly the number of solutions of equation (4) also increases.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

At the next stage, the adversary finds an expression for determining the number of ways to select primitive elements for one generated system of working bases. The primitive polynomial $\alpha_i(x)$ is selected with a check for primitivity, and its degree must not be higher than the degree m_i of the base $p_i(x), i = 1,2,\dots,S$. The total number of all possible choices of primitive polynomials for all working bases will then be::

$$f = \prod_{i=1}^{s}(2^{m_i} - 2). \quad (5)$$

The final step of the cryptanalyst after the positive results in the two previous stages is to find one of the secret keys of parties *A* and *B*. Since there is only one secret key for the system of bases, then finding a secret key automatically makes it possible to find other primitive elements. At this stage, the number of all possible checks is described by an expression that coincides with (5). Then, taking into account (4) and (5), the cryptographic strength of the modified algorithm is determined by the expression:

$$p_{cg} = \frac{1}{\sum_{k_1 k_2 \dots k_S}\left[(k_1+k_2+\dots+k_s)!c_{n_1}^{k_1}c_{n_2}^{k_2}\dots c_{n_S}^{k_S}\left(\prod_{i=1}^{S}(2^{m_i}-2)\right)^2\right]}. \quad (6)$$

It follows from expression (6) that a cryptanalyst needs to solve a very difficult problem to find one of the secret keys when they are openly distributed.

Let us determine the values of the cryptographic strength p_{cg} of the public key distribution algorithm developed on the basis of modular arithmetic for key lengths of 128, 192, and 256 bits of the AES encryption standard.

For the secret key length $G_{cg}(x)$ = 128 bits, we will form a system of working bases of 8 irreducible polynomials of degree 16, then $p_{cg} \approx \frac{1}{8!C_{4080}^{8}((2^{16}-2)^8)^2} = \frac{1}{10^{106}}$, where 4,080 is the number of all irreducible polynomials of degree 16 over the field GF (2).

For a secret key of 192 bits, when choosing a system of working bases of 12 irreducible polynomials of degree 16, we have $p_{cg} \approx \frac{1}{10^{159}}$.

With a key length of 256 bits for the selected base system, out of 16 irreducible polynomials of degree 16, we get $p_{cg} \approx \frac{1}{10^{212}}$.

As can be seen from the above calculations, the use of NPNs (or an unconventional approach) in the construction of asymmetric unconventional algorithms for open exchange of secret keys and encryption can significantly increase their cryptographic strength. The efficiency of these algorithms lies in parallel modular calculations performed with the use of properties of operations in NPNs.

CONCLUSION

The development of key distribution algorithms using residue number systems has the following advantages:

– Inter-digit carry free addition;

– Relatively small numbers used in modular operations;

– The ability to detect and correct errors;

– The possibility of developing highly effective means of protecting information in various electronic systems and networks.

The algorithm we examined can solve some of the key management issues described above. The next stages of the work will be a study of the security of the proposed model and its software implementation. Testing and implementation of the key management model in cryptographic systems will also be conducted.

ACKNOWLEDGMENTS

This work was carried out as part of the grant financing program AP05132568 "Development of a cryptographic key management system" of the Ministry of Education and Science of the Republic of Kazakhstan.

REFRENCES:

[1] Fomina I.A. Key management in cryptographic systems. Bulletin of the Lobachevsky State University of Nizhny Novgorod, Series Informatics, 2010, No. 4(1), P. 165-169.

[2] Schneier B. Applied Cryptography. Protocols, Algorithms, and Source Code in C. Moscow, Publishing house Triumph, 2003, 816 p.

[3] Barker E. Recommendation for Key Management – Part 1: General, NIST Special Publication 800-57, Revision 4, 2016. 160 p.

[4] Barker E., Smid M., Branstad D., Chokhani S. A Framework for Designing Cryptographic Key Management Systems / NIST Special Publication 800-130, Revision 4,.2013. 120 p.

[5] Kapalova N.A., Abisheva A.Zh. Centralized cryptographic key management system. Proc. of IV International research-to-practice conf. "Informatics and Applied Mathematics". Almaty, 2019. P. 569-575.

[6] Varennikov A.V. A brief overview of the basic principles of the development of cryptographic key management systems. Proc. of Scientific Conference "Modern problems of informatics and computational technologies." Almaty, 2019. P. 154-160.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[7] Diffie W. and Hellman M. New directions in cryptography. IEEE Transactions on Information Theory. 1976, Vol.1, T.22. P.644-654.

[8] [X.509-00] ITU-T. Recommendation X.509: The Directory — Authentication Framework, 2000, https://www.itu.int/rec/T-REC-X.509: 05.2019.

[9] Fiat A. and Shamir A. How to prove yourself: Practical solutions to identification and signature problems. Proc. of CRYPTO86, Advances in Cryptology. Springer-Verlag, 1987. P.186-194.

[10] Shamir A. Identity-based cryptosystems and signature schemes. Proc. of CRYPT0'84. Advances in Cryptology. Springer-Verlag, 1985. P.47-53.

[11] Guillou LC. and Quisquater J.J. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. Proc. of EUROCRYPT88. Advances in Cryptology. Springer-Verlag, 1988. P. 123-128.

[12] Paillts J.C. and Girault M. CRIPT: A public-key based solution for secure data communications. Proc. of SECURICOM, 1989. P.171-185.

[13] Girault M. and Pailles J.C. An identity-based identification scheme providing zero-knowledge authentication and authenticated key exchange. Proc. of ESORICS, 1990. P.173-184.

[14] Girault M. An identity-based identification scheme based on discrete logarithms modulo a composite number. Proc. of EUROCRYPT90, Springer-Verlag, 1991. P.481-486.

[15] Kapalova N.A. A study of the cryptographic strength of the ElGamal encryption algorithm based on non-positional polynomial notations. Information Security. Proc. of XII International research-to-practice conf., Taganrog, SFUTTI, 2012, Part 1. P. 253-258.

[16] Kapalova N.A. A modified ElGamal encryption algorithm based on non-positional polynomial notations. Proceedings of the National Academy of Sciences of the Republic of Kazakhstan. Almaty, 2013, No. 1. P. 22-26.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

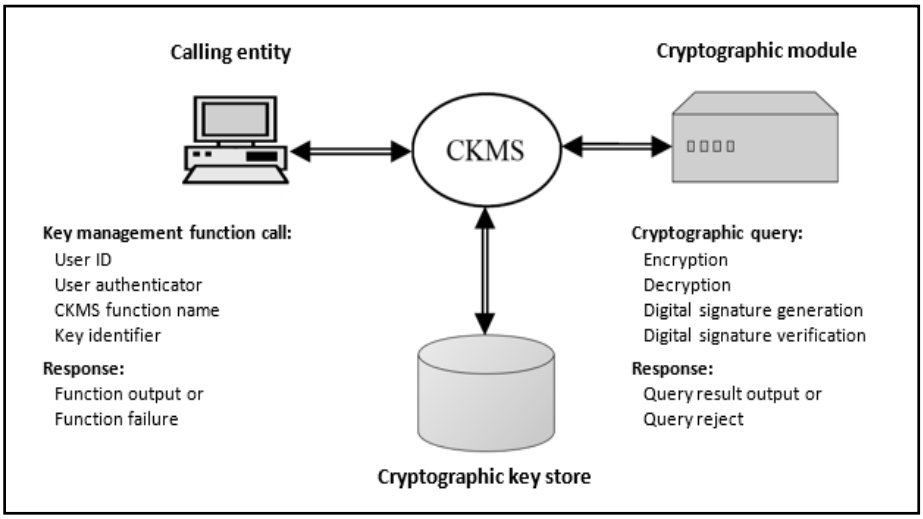

Figure 1: The logic diagram of the key management process

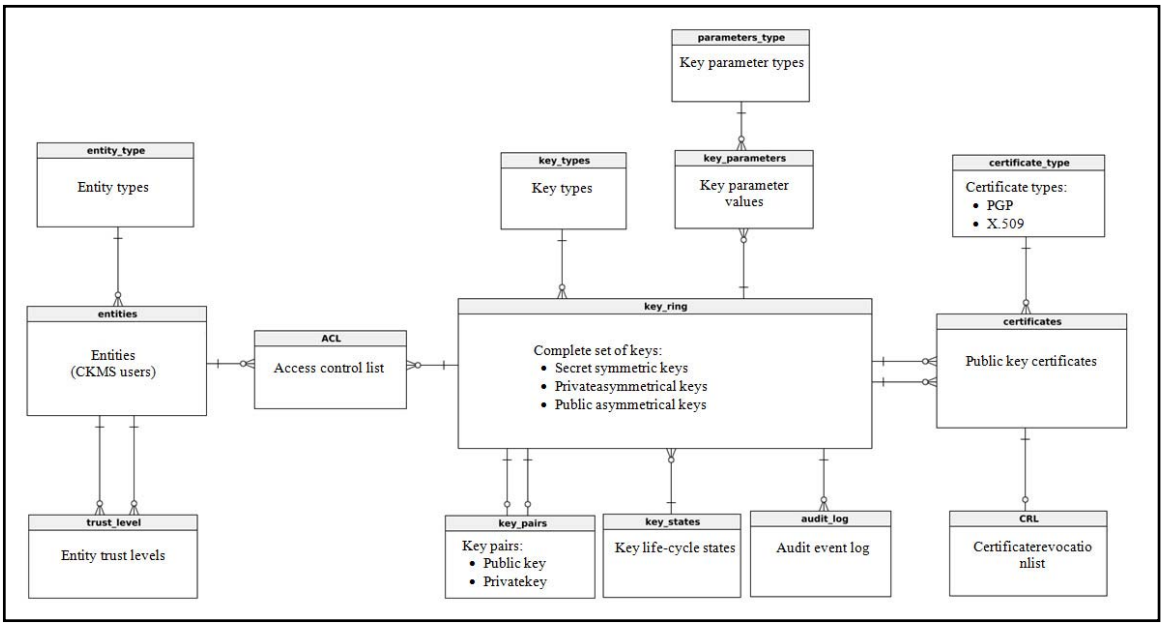

Figure 2: The logical structure of the cryptographic key store

Journal of Theoretical and Applied Information Technology
15th November 2020. Vol.98. No 21

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

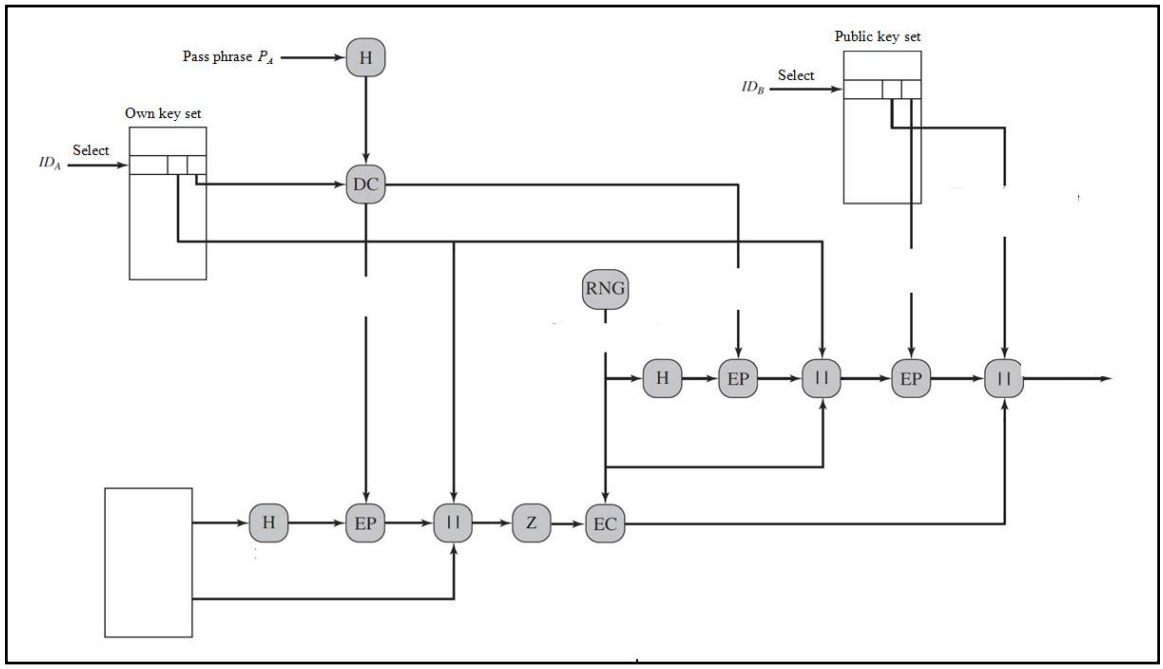

Figure 3: Secure message transmission scheme

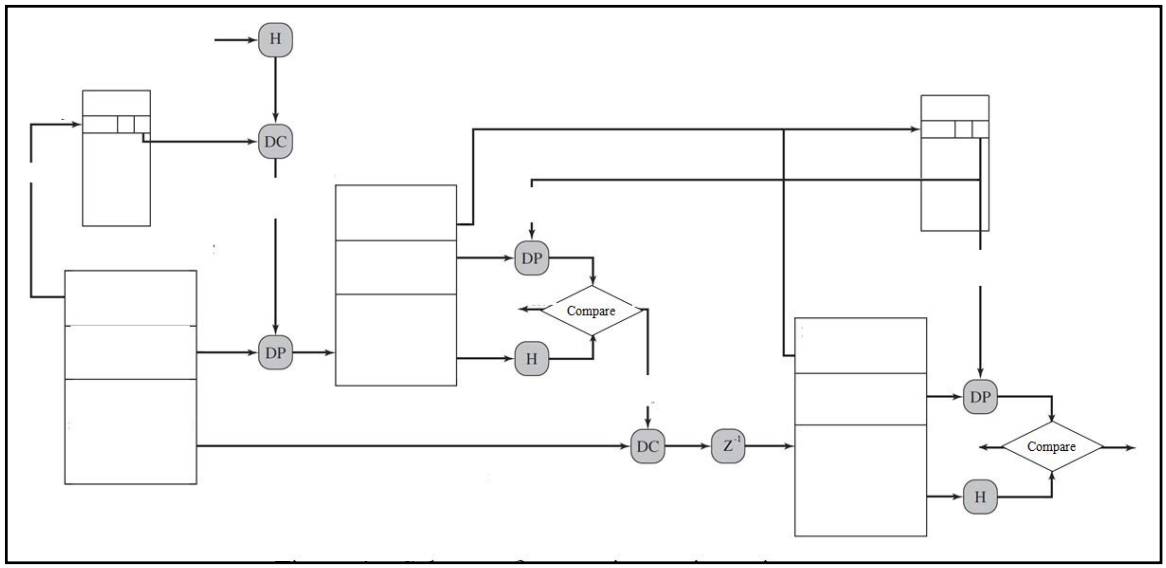

Figure 4: Scheme of processing an incoming message

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

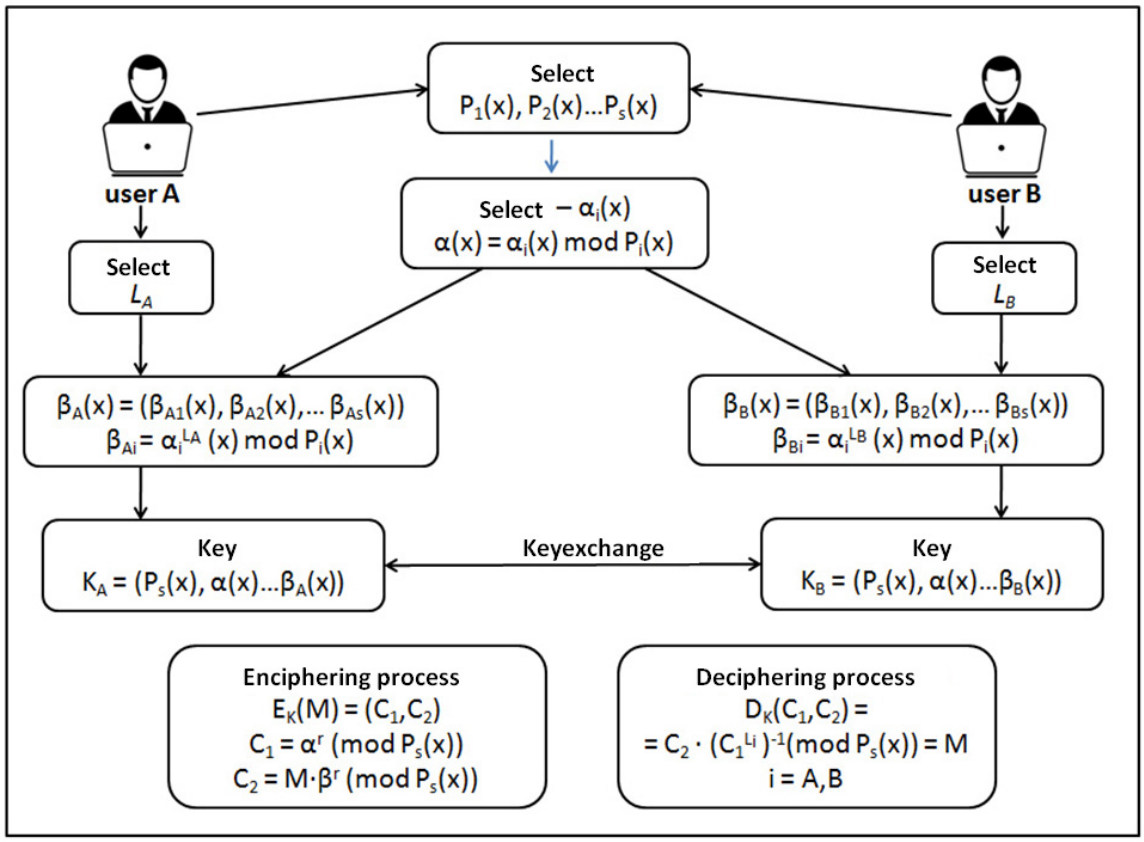

Figure 5: The flowchart of modified asymmetric encryption as per the ElGamal scheme