Comparative Study of the Characteristics of Modern Asymmetric Encryptions

Trung T. Pham[1]

[1]Centro de Investigación en Tecnología de Información
Universidad de Talca – Campus Lircay – Talca – Chile

tpham@utalca.cl

Abstract. *This paper presents a comparative study of five common asymmetric encryption algorithms with the objective of identifying one that is appropriate for satisfying a set of requirements by software developers normally working in a client-server environment. The five modern asymmetric algorithms within the scope of this paper are identified through reviews of current software. The performance indicators are derived through the basic definition of what is a good encryption algorithm, and the data for these indicators are obtained through literature review and computer simulations. Quantitative analysis method of selecting an encryption is presented, and applied to these five common algorithms, with data of their performance indicators and the user's preferential weights, to demonstrate the rationale in making a decision (of selection).*

1. Introduction

Encryption, i.e., Cozzen and Miller (2013), Loshin (2013); is the process of encoding raw (original) data into a coded format so that nobody can understand anything when viewing the converted results. This process has been used primarily for protecting confidential data from being viewed by unauthorized third parties, i.e., Sloan and Warner (2013), Congram et al (2013), but can also be used for protecting the integrity of the data, i.e., Warkentin (2006), Chin and Older (2010), authenticating the genuineness of the data, i.e., Smith (2001), Todorov (2007), and preserving the identity of the creator of the data, i.e., Zhou (2001), Onieva and Zhou (2010). Whenever encryption is mentioned, it is implicitly assumed that decryption, the reverse process of converting encrypted data back to its original format, exists and is available to the appropriate party to access the original data as intended.

Encryption has been used in many software applications, and became even more common when the applications involve data communications across the Internet where data are routed through many transmitting independent servers. These servers, while in theory must agree to follow some security conventions, i.e. Medhi (2007), Sportack (1999), in reality can easily provide hackers or malicious parties a point of unauthorized access, i.e., Sloan and Warner (2013), Allsopp (2009). In this situation, encrypting data is a common sense solution to combat unauthorized access of data while in transit. With the proliferation of many encryption software applications on the market, it can be puzzling even for a programmer to decide which encryption algorithm to use. To provide more choices to the users, these software applications often include both symmetric encryption and asymmetric encryption. According to basic definition, symmetric encryption, e.g., Elminaam et al (2010), Ramkumar (2014), is the encryption that depends on a single secret key for both encrypting and decrypting data and is

normally used for a single user to protect a single computer system, and asymmetric encryption, i.e., Agoyi and Seral (2010), Heuer, Jager, Schäge, and Kiltz (2016),is the encryption that depends on a pair of keys: a public key for everybody to use to encrypt data and a private key for only a single party to decrypt data, and is normally used for an environment with many parties needing to encrypt data for one recipient to decrypt.

Since the introduction of the World Wide Web, the client-server model has been commonly used for software to maintain and distribute data, e.g., Morville and Rosenfeld (2006), Sebesta (2012). In this model, a server maintains a depository of data and distributes selected data to clients upon their requests. As the need for securing data in the communications between the server and clients increases, asymmetric encryption and decryption were implemented into the Transmission Control Protocols, i.e., Cerf, Dalal, and Sunshine (2014), Fall and Stevens (2011), for the transmission of data on the Internet. In this protocol, the recipient would issue a public key to the sender for encrypting the data before sending it out. The recipient, upon receiving the encrypted data, will use a private key to decrypt the data back to its original form. The selection of an encryption algorithm is done by the administration of a computer system.

In this paper, a systematic method of evaluating the asymmetric encryption algorithms is developed according to the general standard definition of what is a strong encryption. This evaluation model is important for programmers who must develop application software utilizing a client-server model to decide what is best for a specific application. While some of the data supporting the evaluation model can be found at various sources, some other performance data are not readily available to allow a complete evaluation involving every criterion in the definition of a good encryption. Thus, simulation is developed in this paper to provide the missing data in the evaluation model where each programmer can adjust the weight of each criterion to configure the evaluation formula according to the specific need of the software in their scope.

2. Definition of a Good Encryption

In a general sense, an encryption algorithm should be evaluated on its merit (to be good or bad) so that the users can decide if it satisfies their need for security in protecting the data. However, the merit of being good or bad can sometimes be subjective and might not be uniformly scaled for comparison purpose. Generally, a definition of a good encryption, i.e., Swenson (2008), Joux (2009), was originally defined as: (i) based on sound mathematics, (ii) been analyzed by competent experts and found to be sound, and (iii) stood the “test of time.”

While the three characteristics of a good encryption can be used as a guideline to evaluate an encryption, they do not give any suggestion to a uniform scale for comparative purpose. With the exception of the number of brute-force attempts required to break an encryption that represents a uniform scale but lacks accuracy in ignoring additional knowledge about the context of the data and about the mathematical algorithm, the remaining two characteristics of using experts to test over a long period of time are vague to the point that they remain as timeless suggestion not relying on any specific technique or technology at any point in time. Thus, these three characteristics are good guidance but lack the specificity of a uniform scale for comparative evaluation that is much needed to decide on a selection. For this reason, a set of characteristics known as Shannon Characteristics, by Shannon (1949) and Shannon (1948), is proposed

Table 1. Summary of the Characteristics of a Good Encryption.

Characteristic Name	Definition Attribute	Objective Measurement	Direct Measurement
Based on Sound Mathematics	Common	YES	YES
Accepted by Experts	Common	YES	NO
Survives the "Test of Time"	Common	YES	YES
Secrecy vs. Effort	Shannon	YES	YES
Complexity in Algorithm	Shannon	NO	NO
Simplicity in Implementation	Shannon	NO	NO
Efficiency in Data Representation	Shannon	YES	YES

to define a good encryption as follows: (i) the amount of secrecy should only be proportional to the efforts of encryption and decryption, (ii) the keys and the algorithms should be free of complexity, (iii) the implementation of the encryption and decryption should be as simple as possible, and (iv) the size of the encrypted data should not be larger than the size of the original data. In these Shannon characteristics, the size of the encrypted data is considered for the first time as an important determining factor of a good encryption.

Table 1 summarizes the seven basic characteristics that define a good encryption based on the common definition and the Shannon definition. In this table, the first three characteristics are from the common definition, and the last four are from the Shannon definition. All except two (complexity in algorithm, and simplicity in implementation) are supported with measurable data. The acceptance by experts is not directly measurable, but is also supported by indirect measurement. These characteristics will be used in the quantitative model of decision for selecting an appropriate encryption algorithm. The data supporting the five characteristics in Table 1 will be extracted from available literature and summarized in this quantitative model. The missing data supporting the two characteristics complexity in algorithm and simplicity in implementation will be measured in computer simulations and also introduced into the quantitative model of decision. The user of this model of decision must assign the weight to represent personal preference to each characteristic so that numerical scores can be calculated for a comparative analysis toward the final selection.

3. Common Asymmetric Encryptions and Their Characteristics

The five asymmetric encryptions selected for this study, based on their popularity in use at current time, are Diffie-Hellman by Hellman, Diffie, and Merkle (1980), RSA by Rivest, Shamir, and Adleman (1978), Paillier by Paillier (1999), Cramer-Shoup by Cramer and Shoup (1998), and ElGamal by ElGamal (1985). In this section, each algorithm is mathematically described so that their unavailable Shannon characteristics (simplicity in implementation, and complexity in algorithm) can be evaluated objectively.

3.1. Diffie-Hellman Algorithm

Diffie-Hellman is a method of generating a common key for two parties, not necessarily trusting each other, to jointly create and share a (temporary) secret key used for encrypting data for exchange. The method allows open communications between the two parties to create a common key, but prevents a third party from re-creating that common key based on the exchanged information in the open communications. In this aspect, the method can be seen as a pseudo-encryption that modifies a conventional symmetric encryption by attaching a key sharing method so that the symmetric encryption can be used in the manner of an asymmetric encryption. The method was first documented in 1976, and is still in use today after 38 years in existence. The method, originally protected under the US Patent 4,200,770 granted in 1977, is now available to the public because the patent protection has expired. The strength of the method is the security of exchanging the same key between two parties in the open (Internet). Additionally, the strengths include the computational efficiency and expansion ratio (advantages of a symmetric encryption over asymmetric encryption). The weaknesses of the method include the shared key between two parties that increase the number possible points of attack, and the additional man-in-the-middle attack where a third party can pretend to be one point of the communication and obtain a secret key and the secret code to generate the key from others.

The Diffie-Hellman algorithm part for generating the encryption keys that can be exchanged securely in public consists of the following steps:

(i) Both Sender 1 and Sender 2 publicly select a prime number p and a base number g (both can be seen by third parties)
(ii) Sender 1 select a secret number a, and sends $A = (g^a \bmod p)$ to Sender 2
(iii) Sender 2 select a secret number b, and sends $B = (g^b \bmod p)$ to Sender 1
(iv) Third parties can intercept $A = (g^a \bmod p)$, $B = (g^b \bmod p)$, p, and g
(v) Sender 1 receives $B = (g^b \bmod p)$, and privately calculates the common secret key $s = (B^a \bmod p)$
(vi) Sender 2 receives $A = (g^b \bmod p)$, and privately calculates the common secret key $s = (A^b \bmod p)$, this is exactly what Sender 1 calculates in $s = (B^a \bmod p)$
(vii) Third parties can set up equation $(B^a \bmod p) = (A^b \bmod p)$ with known A, B, p, and g, but cannot solve for a and b and therefore cannot calculate the private key s.

3.2. RSA Algorithm

RSA is a method of asymmetric encryption with the generation of a public key based on the multiplicative product of two large prime factors. The method was first documented in 1977, and is still in use today after 37 years in existence. The method was originally protected by the US Patent 4,405,829 granted in 1983, is now available to the public because the patent protection has expired. This method has two components: generating a public key, and performing encryption. The method of generating public key has the advantage over the Diffie-Hellman in minimizing the number of locations that maintain the secret key (one vs. many). The encryption algorithm is less computational efficient, and has a slightly larger expansion ratio than the Diffie-Hellman (disadvantages of asymmetric encryption over symmetric encryption). However, the method is based on

the multiplication operator and therefore is more computationally efficient than the Paillier, ElGamal, and Cramer-Shoup encryptions. Similarly, the expansion factor 1:1 of the RSA is better than the Paillier, ElGamal, and Cramer-Shoup encryptions.

The RSA algorithm part for generating two encryption keys that will be used for encrypting and decrypting consists of the following steps:

(i) Choose two prime numbers p and q of similar bit lengths (use Primality Test to verify p and q are primes)

(ii) Compute $n = pq$, where n will be used as the modulus for both the public and private keys. The length of n (in number of bits) is called the key length.

(iii) Compute the Euler's totient function $\varphi(n) = (p-1)(q-1) = n-(p+q-1)$

(iv) Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$; i.e., e and $\phi(n)$ are coprime
- e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings
- e is released as the public key exponent

(v) Determine d as $d \equiv e^{-1} \pmod{\phi(n)}$; i.e., d is the multiplicative inverse of e (modulo $\phi(n)$).
- solve for d when given $d \cdot e \equiv 1 \pmod{\phi(n)}$ by using the extended Euclidean algorithm.
- d is kept as the private key exponent

(vi) The *public key* consists of the modulus n and the public (or encryption) exponent e. The *private key* consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. The selected prime numbers p, q, and their corresponding Euler's totient function $\phi(n)$ must also be kept secret because they can be used to calculate d.

The RSA algorithm part for encrypting data with a public key and decrypting data with a private key consists of the following steps

(i) The public key (n, e) is transmitted freely to whoever wants to send encrypted message

(ii) The message M is converted into an integer m, such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme

(iii) The cipher text c is calculated with the public key (n,e) as $c \equiv m^e \pmod{n}$

(iv) The cipher text c is decrypted back with the private key (n,d) to the message $m^* \equiv c^d \pmod{n}$

3.3. Paillier Algorithm

Paillier encryption is a method of asymmetric encryption based on a probabilistic model. The method was first documented in 1999. The method was originally protected under the US Patent 7,054,444 granted in 2006, and is expected to expire in 2026. The encryption algorithm is less computational efficient, and has larger expansion ratio (2:1) than the Diffie-Hellman (disadvantage of the probabilistic model over the simple multiplication model).

The Paillier algorithm part for generating two encryption keys that will be used for encrypting and decrypting consists of the following steps:

(i) Choose two large prime numbers p and q randomly and independently of each other such that gcd(pq, $(p-1)(q-1)$) = 1. This property should be assured if both primes are of equivalent length.
(ii) Calculate $n = pq$, and $\lambda = \text{lcm}(p-1, q-1)$.
(iii) Select random integer g where $g \in Z^*_{n^2}$ (a notation for all integers less than n^2 and greater than zero: $g \in \{x \mid x \in Z \text{ and } 0 < x < n^2\}$)
(iv) Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \bmod n^2)^{-1} \bmod n$, where the function $L(\cdot)$ is defined as $L(u) = (u-1)/n$. Note that the notation a/b does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of a divided by b, i.e., the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.
(v) The *public key* consists of the modulus n and the random integer g.
(vi) The *private key* consists of the integers λ and μ.

The Paillier algorithm part for encrypting data with a public key and decrypting data with a private key consists of the following steps:

(i) The public key (n, g) is transmitted freely to whoever wants to send encrypted message
(ii) The message M is converted into an integer m where $m \in Z_n$.
(iii) Select a random integer $r \in Z^*_n$ or $r \in \{1, 2, \ldots, n\}$
(iv) The cipher text c is calculated with the public key (n,g) as $c \equiv g^m \cdot r^n \pmod{n^2}$
(v) The cipher text c is decrypted back with the private key (λ,μ) to the message $m^* = L(c^\lambda \bmod n^2) \cdot \mu \pmod{n}$

3.3. ElGamal Algorithm

ElGamal algorithm is a method of encryption based on the discrete logarithms. The method was first documented in 1985. The method was not protected under any US Patent and therefore is available to the public. The encryption algorithm is less computational efficient than the Paillier encryption (disadvantage of the discrete logarithmic model over the probabilistic model). The expansion ratio of 2:1 is the same as that of the Paillier encryption.

The ElGamal algorithm part for generating two encryption keys that will be used for encrypting and decrypting consists of the following steps:

(i) Choose an integer q and generate a cyclic group G of order q with the element of generator g:
$$G = \{ g^1, g^2, \ldots, g^{q-1} \}$$
(ii) Choose a random $x \in \{1, 2, \ldots, q-1\}$
(iii) Calculate $h = g^x$
(iv) The *public key* consists of the constant h, the description of the cyclic group G, q, and g.
(v) The *private key* consists of just the integers x.

The ElGamal algorithm part for encrypting data with a public key and decrypting data with a private key consists of the following steps:

(i) The public key (h, q, g, G) is transmitted freely to whoever wants to send encrypted message
(ii) The message M is converted into an integer m
(iii) Choose a random y from { 1, 2, …, q – 1 }, then calculate $c_1 = g^y$
(iv) Calculate the shared secret $s = h^y$
(v) Convert the message m into an element m' of G.
(vi) Calculate $c_2 = m' \cdot s = m' \cdot h^y$
(vii) The cipher text is $(c_1,c_2) = (g^y, m' \cdot h^y) = (g^y, m' \cdot (g^x)^y)$
(viii) The cipher text (c_1,c_2) is decrypted back with the private key x, with the shared secret $s = c_1^x$, and the decrypted message $m^* = c_2 \cdot s^{-1}$

3.5. Cramer-Shoup Algorithm

Cramer-Shoup encryption is a method of encryption based on the extension of the ElGamal encryption. The method was designed to prevent the malleability that exists in the ElGamal encryption. The method was first introduced in 1998. The method was originally protected by the US Patent 6,697,488 granted in 2004, and is expected to expire in 2024. The encryption algorithm is less computational efficient than the ElGamal encryption (because of additional modification to prevent the problem of malleability). The expansion ratio of 4:1 is worse than those of other encryption algorithms.

The Cramer-Shoup algorithm part for generating two encryption keys that will be used for encrypting and decrypting consists of the following steps:

(i) Choose an integer q and generate a cyclic group G of order q with the element of generator g
(ii) Choose five random values x_1, x_2, y_1, y_2, z from { 0, 1, 2, …, q – 1 }. These five values { x_1, x_2, y_1, y_2, z } will be retained as secret key.
(iii) Calculate $c = g_1^{x_1} g_2^{x_2} \bmod q$, $d = g_1^{y_1} g_2^{y_2} \bmod q$, and $h = g_1^z \bmod q$. These three values { c, d, h } will be the public key.

Table 2.Indicators of a Good Encryption with Flexibility for Modification and Implementation.

Indicator	Notation	Method of Measurement
Based on Sound Mathematics	x_1	Encryption algorithm is published in reputable peer-reviewed journals
Accepted by Experts	x_2	Encryption has been implemented in commercial applications
Survives "Test of Time"	x_3	The number of years that encryption has been in use up to now
Secrecy vs. Effort	x_4	The computational resources required to generate the key and to encrypt the data
Complexity in Algorithm	x_5	The number of mathematic operations required in the formula of the algorithm
Simplicity in Implementation	x_6	The number of programming instructions required in the program where the algorithm is implemented
Efficiency in Data Representation	x_7	Expansion ratio measuring the size (or length) of encrypted data in comparison to the size of the original unencrypted data
Flexibility for Modification	x_8	Not under protection of patent
Computational Efficiency	x_9	The time required to execute a program generating keys, encrypting data, and decrypting data

The Cramer-Shoup algorithm part for encrypting data with a public key and decrypting data with a private key consists of the following steps:

(i) The public key (c, d, h) is transmitted freely to whoever wants to send encrypted message

(ii) Convert the original message M into $m \in \{ 1, 2, \ldots, q-1 \}$

(iii) Choose a random $k \in \{ 1, 2, \ldots, q-1 \}$, and calculate $u_1 = g_1^k \bmod q$, and $u_2 = g_2^k \bmod q$

(iv) Calculate $e = (h^k \cdot m) \bmod q$

(v) Calculate $\alpha = H(u_1, u_2, e)$ where $H(\cdot,\cdot,\cdot)$ is a universal one way hash function (or one can use a collision resistant hash function) (this is part of the validity check)

(vi) Calculate $v = c^k d^{k\alpha} \bmod q$

(vii) The cipher text consists of (u_1, u_2, e, v)

(viii) The cipher text (u_1, u_2, e, v) is validated with $\alpha = H(u_1, u_2, e)$ through the verification of the equality $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha} = v$, this equality must be verified before proceeding to the decryption $m^* = e / (u_1^z)$

4. Quantitative Decision Model for Evaluation and Selection

In the context of this study of selecting the best suitable encryption algorithm out of the already identified five asymmetric encryption algorithms, the selection process can be

formed as a discrete optimization problem, with an objective function defined by the criteria of selection. The objective function is in the form of an analytical function with dependent variables defined with numerical values. All available options are evaluated and the option with the best evaluation is selected.

From Table 1, the seven characteristics defining a good encryption are used as indicators of a good encryption. However, from a programmer's perspective, the flexibility for modification and computational efficiency are important in the development of software application, and therefore are added to the list for a total of nine indicators shown in Table 2. In this table, the definition for each indicator is included so that an objective function $J(x_1, x_2, \ldots, x_9)$ can be meaningfully formulated to reflect consideration for each of the indicator. When all indicators are considered in the same manner, the weighted combination is often used in the following:

$$J(x_1, x_2, \ldots, x_9) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_9 x_9,$$

where the constants $\alpha_1, \alpha_2, \ldots, \alpha_9$, are weight constants, with normalized values between 0 and 1, bearing the preference of a programmer toward a specific variable: the lower value means less preference and the higher value means more preference. The weight constants $\alpha_1, \alpha_2, \ldots, \alpha_9$ must be defined by the decision making person who decides what variable carries more weight (more importance) than other variables. When it is not clear to a user what variable should carry more weight, it is common to assign equal weights to all variable. For understanding the difference in how a variable might affect the selection, it is common to vary the weights of the variables to see how the selection changes with the changing of the weights.

For each encryption algorithm, the values for the variables $x_1, x_2, \ldots, x_9$ can be obtained through the lookup tables listed from Table 3. Appendix 1 provides more details on how these values are derived. These values are fed into the objective function $J(x_1, x_2, \ldots, x_9)$, along with the user-defined constants $\alpha_1, \alpha_2, \ldots, \alpha_9$, and the value for the objective function $J(x_1, x_2, \ldots, x_9)$ is calculated. The results for all algorithms are tabulated, with higher value meaning better and lower value meaning worse. Table 4 show some examples of the calculation of $J(x_1, x_2, \ldots, x_9)$ for the five algorithms with various combination of the weight constants $\alpha_1, \alpha_2, \ldots, \alpha_9$. In the first line, all weight constants are set to be equal. In the other lines, all weight constants are set to be equal with the exception of one specific constant (mentioned in the first column) being set to have double the value of the other constants. This table is being shown for illustrative purpose. The users should indicate their own personal preference for placing the importance on each variable through the setting of the corresponding weight constant, with higher value meaning more importance, and lower value meaning less importance.

5. Conclusion

The five algorithms Diffie-Hellman, RSA, Paillier, ElGamal, and Cramer-Shoup have been identified as the commonly used asymmetric encryption. The definition of what is a good encryption is analyzed and transformed into measureable indicators that are used as variables in an optimization problem representing a decision making process of selecting a good encryption. In this decision model, the values of the variables are provided through review of literature and computer simulations, and the weights (of emphasis) of the variables are identified by the users according to their specific need.

The final numerical results are used as guidance to comparative evaluation and selection of an appropriate one.

The values of the nine variables listed in Table 2 can be obtained for the decision model based on an optimization problem with the objective function $J(\cdot)$ mentioned earlier. In this endeavor, the values are either from available literature, analysis of the encryption formula, or from computer simulations. The values for these variables are listed in the following Tables 5 - 11. In Table 5 for the variable x_5, it is assumed that 64-bit calculation is used, and the value for each addition or subtraction operation can be assigned to be 64, and subsequently the value for each multiplication or division operation will be 64^2. When the total value is tallied, higher value means more computational resources and lower value means less computational resources. To convert this value, denoted as v_n, for $n = 1, 2, \ldots, N$, to the normalized value where higher value means better and lower value means worse, one needs to perform the normalization calculation $w_n = 1 - v_n / \max\{v_1, v_2, \ldots, v_N\}$. This normalization formula can also be applied to other variables x_3, x_4, x_6, x_7, and x_9. For variables x_1, x_2, and x_8, the description "YES" will be assigned the value 1 and the description "NO" will be assigned the value 0. For variables x_4 and x_9, computer simulation is used to evaluate the computational performance. In this simulation, each algorithm is coded and executed for a large number of times, and the execution time is estimated by subtracting the initial time from the end time of the program. Since this is an estimation, the simulation is repeated and the average is calculated. It is important to use the same computer for all simulations without any additional program running in the background that can potentially interfere with the measurement of time. Note that all source codes for the simulation can be found in the website listed in the section of supplemental materials.

References

Agoyi, M., and Seral, D. (2010). "SMS Security: An Asymmetric Encryption Approach." Proceedings of the IEEE 6th International Conference on Wireless and Mobile Communications, Valencia, pp. 448-452.

Allsopp, W. (2009). Unauthorised Access: Physical Penetration Testing For IT Security Teams. West Sussex, UK: John Wiley & Sons.

Cerf, V., Dalal, Y., and Sunshine, C. (2014). Specification of Internet transmission control program. Seattle, WA: Amazon Digital Services.

Chin, S. K. and Older, S. B. (2010). Access control, security, and trust: a logical approach. Boca Raton, FL: Chapman and Hall/CRC.

Congram, M., Bell, P., and Lauchs, M. (2013). Policing Transnational Organized Crime and Corruption: Exploring the Role of Communication Interception Technology. New York, NY: Palgrave McMillan.

Cozzens, M. and Miller, S. J. (2013). The Mathematics of Encryption: an Elementary Introduction. Providence, RI: The American Mathematical Society.

Cramer, R., and Shoup, V. (1998). "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack." Proceedings of CRYPTO-98, LNCS 1462, pp. 13-25.

ElGamal, T. (July, 1985). “A public key cryptosystem and a signature scheme based on discrete logarithms.” IEEE transactions on information theory, vol. 31, no. 4, pp 469-472.

Elminaam, D. S. A., Kader, H. M. A., and Hadhoud, M. M. (May 2010). “Evaluating the Performance of Symmetric Encryption Algorithms,” International Journal of Network Security, vol. 10, no. 3, pp. 213–219.

Fall, K. R., and Stevens, W. R. (2011). TCP/IP illustrated, Volume 1: the Protocols. Boston, MA: Addison-Wesley Professional.

Hellman, M. E., Diffie, B. W., and Merkle, R. C. (April 29, 1980). “Cryptographic apparatus and method.” US Patent US4200770A. Washington, DC: US Patent and Trademarks Office.

Heuer, F., Jager, T., Schäge, S., and Kiltz, E. (November 2016). “Selective opening security of practical public-key encryption schemes.” IET Journal of Information Security, vol. 10, no. 6, pp. 304-318.

Joux, A. (2009). Algorithmic cryptanalysis. Boca Raton, FL: Chapman and Hall/CRC.

Loshin, P. (2013). Simple Steps to Data Encryption: a Practical Guide to Secure Computing. Waltham, MA: Syngress/Elsevier.

Medhi, D. (2007). Network Routing: Algorithms, Protocols, and Architectures. San Francisco, CA: Morgan Kaufmann.

Morville, P., and Rosenfeld, L. (2006). Information architecture for the World Wide Web: designing large-scale Web sites. Sebastopol, CA: O'Reilly Media.

Onieva, J. A. and Zhou, J. (2010). Secure multi-party non-repudiation protocols and applications. New York, NY: Springer.

Paillier. P. (1999). “Public-key cryptosystems based on composite degree residuosity classes.” EUROCRYPT: Lecture notes in computer science, vol. 1592, pp. 223–238.

Ramkumar, M. (2014). Symmetric Cryptographic Protocols. New York, NY: Springer.

Rivest, R. L., Shamir, A., and Adleman, L. M. (1978). “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.” Communications of the ACM, vol. 21, no. 2, pp. 120-126.

Sebesta, R. W (2012). Programming the World Wide Web. Boston, MA: Addison-Wesley.

Shannon, C. E. (1949) “Communication theory of secrecy systems.” Bell System technical journal, vol. 28, no. 4, pp. 656–715.

Shannon, C. E. (1948). “A mathematical theory of communication.” Bell System technical journal, vol. 27, no. 3 (1948) pp. 379-423.

Sloan, R. H. and Warner, R. (2013). Unauthorized Access: the Crisis in Online Privacy and Security. Boca Raton, FL: CRC Press

Smith, R. E. (2001). Authentication: from Passwords to Public Keys. Boston, MA: Addison-Wesley Professional.

Sportack, M. (1999). IP Routing Fundamentals. Indianapolis, IN: Cisco Press.

Swenson, C. (2008). Modern cryptanalysis: techniques for advanced code breaking. Indianapolis, IN: Wiley.

Todorov, D. (2007). Mechanics of User Identification and Authentication: Fundamentals of Identity Management. Boca Raton, FL: Auerbach Publications.

Warkentin, M. (2006). Enterprise Information Systems Assurance and System Security: Managerial and Technical Issues. Hershey, PA: Idea Group Publishing.

Zhou, J. (2001). Non-Repudiation in Electronic Commerce. Boston, MA: Artech House.

Supplemental Material

Source codes for the encryption, decryption, and simulations are listed at http://iie.utalca.cl/citi/projects/encryption

Acknowledgment

This study was supported in part by the Chilean CONICYT Research Grant FONDEF-IDEA CA13I10317 of the project “Authentication of Wine with Electronic Information” awarded for the period of 2014-2015.

Appendix 1. Calculation of Variable Values

Table 3. Numerical Data on a Uniform Scale for Variables Used in an Objective Function of Selecting an Encryption Algorithm (0 means worst, 1 means best).

Variables	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
x_1	1.00	1.00	1.00	1.00	1.00
x_2	1.00	1.00	1.00	1.00	1.00
x_3	1.00	0.97	0.39	0.76	0.42
x_4	0.34	1.00	0.47	0.97	0.28
x_5	0.25	1.00	0.63	0.86	0.22
x_6	0.38	1.00	0.71	0.83	0.42
x_7	1.00	1.00	0.50	0.50	0.25
x_8	1.00	1.00	0.00	1.00	0.00
x_9	0.28	1.00	0.46	0.95	0.25

Table 4. Numerical Evaluation of Normalized Objective Function $J(x_1, x_2, \ldots, x_9)$ (0 means worst, 1 means best).

Weights	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
equal	0.70	1.00	0.58	0.88	0.43
double α_1	0.73	1.00	0.62	0.89	0.49
double α_2	0.73	1.00	0.62	0.89	0.49
double α_3	0.73	1.00	0.56	0.87	0.43
double α_4	0.66	1.00	0.56	0.89	0.41
double α_5	0.65	1.00	0.58	0.88	0.41
double α_6	0.66	1.00	0.59	0.87	0.43
double α_7	0.73	1.00	0.57	0.84	0.41
double α_8	0.73	1.00	0.52	0.89	0.39
double α_9	0.65	1.00	0.56	0.88	0.41

Table 5. Summary of the Computational Count of the Asymmetric Encryption Algorithms, used for variable x_5

	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
Key Generation	0 additions, 0 multiplications, 0 inverse, 4 power, 4 modulus	3 additions, 1 multiplication, 1 inverse, 0 power, 3 modulus	3 additions, 3 multiplications, 1 inverse, 0 power, 1 modulus	0 addition, 0 multiplication, 0 inverse, 1 power, 1 modulus	0 addition, 2 multiplications, 0 inverse, 5 powers, 7 modulus
Encryption	n.a.	0 addition, 0 multiplication, 1 power, 1 modulus	0 addition, 1 multiplication, 2 powers, 3 modulus	0 addition, 1 multiplication, 2 powers, 3 modulus	2 additions, 3 multiplications, 5 powers, 7 modulus
Decryption	n.a.	0 addition, 0 multiplication, 0 division, 1 power, 0 inverse, 1 modulus	2 additions, 0 multiplication, 1 division, 1 power, 0 inverse, 1 modulus	0 addition, 1 multiplication, 0 division, 0 power, 2 inverses, 3 modulus	2 addition, 4 multiplications, 0 division, 5 power, 1 inverse, 11 modulus

Table 6. Summary of the Implementation of Java Codes for the Asymmetric Encryption Algorithms, used for variable x_6.

	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
Key Gen.	4 instructions	3 instructions	4 instructions	1 instruction	3 instructions
Encryption	n.a.	1 instruction	1 instruction	3 instructions	5 instructions
Decryption	n.a.	1 instruction	2 instructions	2 instructions	4 instructions
Total	4 instructions	5 instructions	7 instructions	6 instructions	12 instructions

Table 7. Summary of the Expansion Ratios of the Asymmetric Encryption Algorithms.

	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
Expansion Ratio (x_7)	n.a.	1.00	2.00	2.00	4.00

Table 8. Summary of Other Aspects of the Asymmetric Encryption Algorithms.

	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
Longevity[1] (x_3)	38	37	15	29	16
Patented (x_2)	YES	YES	YES	NO	YES
Protection (x_8)	NO[2]	NO[2]	YES[3]	NO	YES[4]
Published in Journals (x_1)	YES[5]	YES[6]	YES[7]	YES[8]	YES[9]

[1]in terms of number of years of existence, [2]patent already expired, [3]patent expired in 2026, [4]patent expired in 2024, [5,6,7,8,9]see references

Table 9. Normalized Computational Performance of Asymmetric Algorithms in Generating Keys (used for variables x_4 and x_9).

Experiment	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
1	6.81	2.14	1.00	3.25	18.22
2	6.75	2.14	1.00	3.27	18.01
3	6.80	2.15	1.00	3.29	18.07
4	6.42	2.02	1.00	3.09	17.12
5	6.73	2.16	1.00	3.22	18.02
6	6.79	2.14	1.00	3.25	18.04
7	6.79	2.14	1.00	3.23	17.99
8	6.76	2.12	1.00	3.30	17.94
9	6.78	2.13	1.00	3.26	17.86
10	6.76	2.18	1.00	3.23	18.03
11	6.94	2.17	1.00	3.24	18.26
12	6.63	2.08	1.00	3.13	17.47
13	6.73	2.18	1.00	3.23	17.97
14	6.52	2.06	1.00	3.16	18.25
15	6.73	2.11	1.00	3.12	17.23
16	6.77	2.12	1.00	3.28	18.03
17	6.98	2.20	1.00	3.37	18.16
18	6.97	2.17	1.00	3.29	18.08
19	6.91	2.14	1.00	3.26	19.07
20	6.63	2.10	1.00	3.20	17.59
Average	6.76	2.13	1.00	3.23	17.97
$\sqrt{Var}$	0.14	0.04	0.00	0.07	0.41

Table 10. Normalized Computational Performance of Asymmetric Algorithms in Encrypting Data (used for variables x_4 and x_9).

Experiment	Diffie-Hellman	RSA	Paillier	ElGamal	Cramer-Shoup
1	n.a.	1.21	3.31	1.00	1.82
2	n.a.	1.19	3.37	1.00	1.86
3	n.a.	1.18	3.37	1.00	1.86
4	n.a.	1.16	3.33	1.00	1.85
5	n.a.	1.18	3.35	1.00	1.85
6	n.a.	1.18	3.37	1.00	1.89
7	n.a.	1.18	3.36	1.00	1.85
8	n.a.	1.18	3.34	1.00	1.84
9	n.a.	1.19	3.39	1.00	1.86
10	n.a.	1.18	3.37	1.00	1.88
11	n.a.	1.22	3.34	1.00	1.86
12	n.a.	1.17	3.27	1.00	1.85
13	n.a.	1.19	3.29	1.00	1.86
14	n.a.	1.18	3.30	1.00	1.85
15	n.a.	1.19	3.32	1.00	1.89
16	n.a.	1.20	3.33	1.00	1.86
17	n.a.	1.17	3.28	1.00	1.85
18	n.a.	1.17	3.30	1.00	1.85
19	n.a.	1.19	3.33	1.00	1.87
20	n.a.	1.19	3.33	1.00	1.87
Average	n.a.	1.18	3.33	1.00	1.86
$\sqrt{Var}$	n.a.	0.01	0.03	0.00	0.02