International Journal of Computer Science, Communication & Information Technology (CSCIT)

Vol.3 - issue1 - 2016

pp.21-24

ElGamal public-key encryption

Amer Daeri
Zawia University, Faculty of Engineering/ CE Department
Zawia, Libya,
ibnjubair1@yahoo.co.uk

. Amer R. Zerek
Zawia University, Faculty of Engineering/ EE Department
, Zawia, Libya,
anas_az94@yahoo.co.uk

Mohamed A. Abuinjam
Zawia University, Faculty of Engineering/ EE Department
, Zawia, Libya,
Mohamed_A@yahoo.com

Abstract— Object of this paper is to illustrate the principle of Elgamal Public-Key encryption system. One of the biggest problems in cryptography is the distribution of keys. Suppose you live in Libya and want to pass information secretly to your friend in Europe. If you truly want to keep the information secret, you do need to agree on some sort of key that you and he can use to encode / decode messages. But you don't want to keep using the same key, or you will make it easier and easier for others to crack your cipher .

But it's also a pain to get keys to your friend. If you mail them, they might be stolen. If you send them cryptographically, and someone has broken your code, that person will also have the next key. If you have to go to Europe regularly to hand-deliver the next key, that is also expensive. If you hire some courier to deliver the new key, you have to trust the courier.

***Index Terms*— Public-Key encryption, cryptography, encode, decode, cipher.**

I. Introduction

This paper considers one of the most used schemes in cryptographic applications, it is ElGamal encryption technique; it is type of public-key encryption, also referred to as *asymmetric encryption*. In public-key encryption systems each entity **A** has a *public key* **e** and a corresponding *private key* **d**. In secure systems, the task of computing **d** given **e** is computationally infeasible. The public key defines an *encryption transformation* **Ee**, while the private key defines the associated *decryption transformation* **Dd**. Any entity **B** wishing to send a message x to **A** obtains an authentic copy of **A**'s public key **e**, uses the encryption transformation to obtain the cipher-text **c = eK = Ee(*x*)**, and transmits **c** to **A**. To decrypt **c**, **A** applies the decryption transformation to obtain the original message x = **Dd**(**c**). The public key need not be kept secret, and, in fact, may be widely available only its authenticity is required to guarantee that **A** is indeed the only party who knows the corresponding private key. A primary advantage of such systems is that providing authentic public keys is generally easier than distributing secret keys securely, as required in symmetric key systems [1].

II. Basic ElGamal encryption

The ElGamal public-key encryption scheme is based on the intractability of the discrete logarithm problem (DLP), which will be described in this section.

A. *Algorithm Key generation for ElGamal public-key encryption*

Each entity creates a public key and a corresponding private key.

Each entity **A** should do the following:-

1. Generate a large random prime **p** and a generator α of the multiplicative group Zp^* of the integers modulo **p**.
2. Select a random integer **a**, $1 \le a \le p-2$, and compute $\alpha^a \bmod p$.
3. **A**'s public key is (p, α, α^a); **A**'s private key is a.

B. *Algorithm ElGamal public-key encryption*

B encrypts a message x for **A**, which **A** decrypts.
Encryption. **B** should do the following:

- Obtain **A**'s authentic public key (p, α, α^a).
- Represent the message as an integer x in the range $\{0,1,\ldots, p-1\}$.
- Select a random integer k, $1 \le k \le p-2$.
- Compute $y_1 = \alpha^k \bmod p$ and $y_2 = x\,\beta^k \bmod p$, where $\beta = \alpha^a$.
- Send the cipher-text **eK(x , k) = (y₁, y₂)** to **A**

C. *Algorithm ElGamal public-key decryption*

To recover plaintext **x** from **eK** , **A** should do the following:

- Use the private key **a** to compute $\boldsymbol{y_1^{p-1-a}}$ **mod** $\boldsymbol{p}$ (Note:($\boldsymbol{y_1^{p-1-a}} = \boldsymbol{y_1^{-a}} = \boldsymbol{\alpha^{-ak}}$).
- Recover $\boldsymbol{x}$ by computing $\boldsymbol{(y_1^{-a}).y_2}$ **mod** $\boldsymbol{p}$**.**

III. Example of Discrete Logarithm Problem (DLP)

Suppose **p =17** is an odd prime , then the message representation **Zp** ={0,1,…,p-1} is a finite field , here **Zp** ={1,2,…,16}, also given **α =3** is a generator of **Zp*** , where **Zp*** is the set of integers which are relatively prime to p , i.e. , **Zp*** ={α^o mod p, α^1 mod p, …, α^{p-2} mod p}. Then:-

Zp*= {$3^0,3^1,3^2,3^3,3^4,3^5,3^6,3^7,3^8,3^9,3^{10},3^{11},3^{12},3^{13},3^{14},3^{15}$}

= {1, 3, 9, 10, 13, 5, 15, 11, 16,14, 8,7, 4 , 12 ,2 ,6 }

Note 3^{16} mod 17 =1.

You have to notice that , Given any **a**, compute **b ≡ α^a mod p** is easy , for example given **a = 10**, **b ≡ 3^{10} mod 17 = 8** , given any **b**, finding an **a** such that **b ≡ α^a mod p** is difficult , for example given **b=14**, what is **a** ?? , ⇒ By searching the table, **a = 9**.

The problem is, when **p** is large, the table becomes very large [2].

IV. Example of ElGamal encryption with artificially small parameters

A. Key generation

Entity **A** selects the prime **p = 17** and a generator **α = 3** of $\mathbf{Z_p^*}$. **A** chooses the private key **a = 6** and computes **α^a mod p = 15**, also denoted by **β ≡ α^a mod p = 15**.

A's public key is **(p = 17, α = 3, α^a = β = 15).**

B. Encryption

To encrypt a message **x = 11**, **B** selects a random integer **k = 3** and computes $\boldsymbol{y_1 = \alpha^k}$ **mod p = 3^3 mod 17 = 10** and $\boldsymbol{y_2 = x\beta^k}$ **mod p = 11× 15^3 mod 17= 11×9 =14**, where **β= α^a**. So **B** sends **y1=10** and **y2=14**, it means **(10,14)** is the encrypted message.

C. Decryption

To decrypt, **A** computes $\boldsymbol{y_1^{p-1-a}}$ **mod p = 10^{10} mod 17 =2** and recovers **x** by computing **x** = ($\boldsymbol{y_1^{p-1-a}}$ $\boldsymbol{mod\ p}$).$\boldsymbol{y_2}$ **Mod p = 2× 14 mod 17 = 11.**

V. Common system-wide parameters

All entities may elect to use the same prime $\boldsymbol{p}$ and generator **α** , in which case $\boldsymbol{p}$ and **α** need not be published as part of the public key. This results in public keys of smaller sizes. An additional advantage of having a fixed base **α** is that exponentiation can then be expedited via pre-computations using fixed base exponentiation algorithms. A potential disadvantage of common system-wide parameters is that larger moduli $\boldsymbol{p}$ may be warranted [1].

VI. Efficiency of ElGamal encryption

- The encryption process requires two modular exponentiations, namely$\boldsymbol{\alpha^k}$ **mod** $\boldsymbol{p}$ and $(\boldsymbol{\alpha^a})^k$ **mod** $\boldsymbol{p}$**.** These exponentiations can be sped up by selecting random exponent **k** having some additional structure, for example, having low Hamming weights. Care must be taken that the possible number of exponents is large enough to preclude a search via a baby-step giant-step algorithm.

- A disadvantage of ElGamal encryption is that there is a *message expansion* by a factor of 2. That is, the cipher-text is twice as long as the corresponding plaintext.

VII. Randomized encryption

ElGamal encryption is one of many encryption schemes which utilize randomization in the encryption process in order to circumvent some attacks such as small encryption exponent and forward search attack.

The fundamental idea behind randomized encryption techniques is to use randomization to increase the cryptographic security of an encryption process through one or more of the following methods:

- increasing the effective size of the plaintext message space;
- precluding or decreasing the effectiveness of chosen-plaintext attacks by virtue of a one-to-many mapping of plaintext to cipher-text; and
- Precluding or decreasing the effectiveness of statistical attacks by leveling the a priori probability distribution of inputs.

VIII. Security of ElGamal encryption

The ElGamal encryption scheme can be viewed as simply comprising a Diffie- Hellman key exchange to determine a session key$\boldsymbol{\alpha^{ak}}$, and then encrypting the message by multiplication with that session key. For this reason, the security of the ElGamal encryption scheme is said to be *based* on the discrete logarithm problem in $\boldsymbol{Zp^*}$, although such an equivalence has not been proven.

IX. Applications, Implementation

The ElGamal encryption uses in anonymous communication between users, where users already employ newsgroups such as alt.anonymous.messages to send PGP encrypted messages to anonymous receivers.

Receivers in this encryption can have one private key to decrypt messages sent from any one of many Incomparable Public keys , also Interface is similar to original GnuPG interface , since Only a few changes needed to be made in the existing code (ElGamal encryption already exists in GnuPG) [3].

X. Conclusion

ElGamal is Similar use to RSA, different details, patented, The contents of public keys are important in protecting the receiver's anonymity from the sender.

Incomparable Public Keys provide a secure and efficient way of accomplishing receiver anonymity. Furthermore incomparable Public Keys are useful in practice with Key Exchange and P2P systems [4].

References

[1] A. Menezes, P. van Oorschot, S. Vanstone "Handbook of Applied Cryptography", CRC Press, 1996.
[2] Topics in Computer Mathematics temp.wpd NTC 1/23/05.
[3] http://www.cacr.math.uwaterloo.ca/hac.
[4] http://www.cs.princeton.edu/~bwaters/research/.