1 Asymmetric Ciphers

Cryptographic systems rely on keys for encryption and decryption. Traditionally, a single key is required to encrypt and to decrypt. In order for the recipient of the encrypted message to be decrypted by the recipient, the key must also be transmitted. However, sending the key over the channel (normal channel) where the actual message will be sent is insecure. The key must be transmitted on a different and secure channel (key channel)[4]. This secure channel where the key should be transmitted cannot be used for normal transmission because it is costly and sometimes difficult for users to access and use[4]. This begs the question whether it is possible to send encrypted messages in such a way that the key can also be transmitted over the normal (insecure) channel and still achieve secure communication. In this section, we focus on solving this problem by describing the relevant and important work on asymmetric ciphers. Figure 5 describes the flow in asymmetric cryptography. This section will not cover in detail the concept of *digital signatures* which is closely related to the *key distribution problem.*

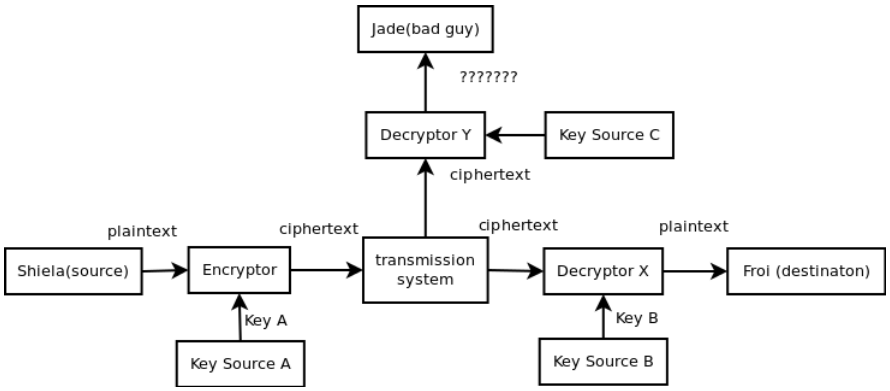

Figure 1: In Asymmetric Cryptography, two keys are used: a public key and a private key. The public key is used for encryption and the private key for decryption. In the figure, if Shiela wants to send a message to Froi, Key A will be Froi's public key and Key B will be Froi's private key. Jade will not be able to decrypt Shiela's message because he does not have Froi's private key. This scheme works because it is computationally difficult for Jade to derive the private key of Froi by just using Froi's public key.

1.1 Merkle (1978)

Secure communication, as described by Merkle[4], allows two parties to communicate in a private manner even though a third party tries its best to learn what is being communicated. We refer to the two parties as Froi and Shiela, and the third party as Jade (Figure 5). Since the key channel is important, the

following describes the characteristics of the channel in relation to Jade.

1. All attempts by Jade to change the messages on the key channel are detectable.

2. Jade will not be able to know the actual content of any message passing on the key channel.

The approach by Merkle relaxes the second condition: It is not necessary for Jade not to know what is being sent in the key channel, he can even know everything passing on it. The challenge then is how to securely distribute the key satisfying the conditions above. If Froi and Shiela have agreed upon a key, and the work needed by Jade to find the key is much higher than the effort by Froi and Shiela needed to generate the key , then it is a solution. The effort by Jade should be exponentially higher compared to the effort by Froi or Shiela for a method to be considered a solution.

Merkle's method uses the concept of puzzles[4]. A puzzle is a cryptogram that is meant to be solved. Any encryption function can be used to generate a puzzle. To allow the puzzle to be solved, the key size (N) used in the encryption function is restricted. The difficulty of solving a puzzle can be controlled by adjusting the size of N. A very large size (in bits) of N will make it very difficult to solve the puzzle. In addition, in order to be able to solve the puzzle, some redundancy is needed. Redundancy is introduced by encrypting, along with the original message, some constant known to Froi, Shiela, and Jade. The absence of the constant when a puzzle is decrypted would mean that a wrong key has been used.

Let us consider the scenario when Shiela wishes to send a message to Froi. First, they both agree on the value of N to use. Shiela then generates N puzzles and transmits these N puzzles to Froi using the key channel. Each puzzle generated will have a puzzle ID and puzzle key. The puzzle ID uniquely identifies each puzzle. The puzzle key on the other hand will be used in future communications that will happen once this puzzle has been solved.

When Froi receives the N puzzles, he selects a puzzle at random and attempts to solve the puzzle, with the amount of effort required, as defined by the size of the key space specified by Shiela. After solving a puzzle, Froi sends the puzzle ID back to Shiela using the key channel. The puzzle key, associated with the puzzle ID sent by Froi, is then used for future communications, this time over the normal channel. At this point Jade knows the puzzle ID, since it was sent using the key channel, but not the puzzle key. If Jade wants to know the key, then he must solve puzzles randomly and check the puzzle ID if it matches the one sent by Froi back to Shiela. This will take Jade a long time to solve. To put it formally, Jade will require $O(N^2)$ effort to determine the key whereas Froi will only need, on the average, $O(N)$. The function below generates the puzzles sent by Shiela to Froi. The encryption function is arbitrary.

```
1  void generate_puzzle()
2  {
```

```
3     bit_string id, key, c, random_key, puzzle, k1, k2;
4     int i;
5
6     k1 = rand(MAXINT);
7     k2 = rand(MAXINT);
8     c = rand(MAXINT);
9     send(c);
10    for (i=0; i<N; i++)
11    {
12        id = encryption_function(k1,i);
13        key = encryption_function(k2, i);
14        random_key = rand(c*N);
15        puzzle = encryption_function(random_key,id,key,c);
16        send(puzzle);
17    }
18 }
```

The code below is executed on Froi's side.

```
1  void get_id()
2  {
3     bit_string id, key, c, selected_puzzle_id, the_puzzle, current_puzzle,
4                temp_constant;
5     int i;
6
7     selected_puzzle_id = rand(N);
8     receive(c);
9     for (i=0; i<N; i++)
10    {
11        receive(currrent_puzzle);
12        if (i == selected_puzzle_id)
13           the_puzzle = current_puzzle;
14    }
15    for (i=0;i<(c*N);i++)
16    {
17        id = get_id(finverse(i, the_puzzle));
18        key = get_key(finverse(i, the_puzzle));
19        temp_constant = get_constant(finverse(i, the_puzzle));
20        if (temp_constant == c)
21           send(id);
22    }
23 }
```

Once Shiela receives the the puzzle ID from Froi, then the following code will be executed. key will be used for subsequent communications between the two.

```
1  void continue_transmission()
```

```
2  {
3      receive(ID);
4      key = encryption_function(k2, ID);
5  }
```

The approach by Merkle requires an effort of $O(N^2)$ from Jade to get the key. However, in todays available computing resources, this can be easily broken. The possibility of exponential methods will be more attractive. Also the amount of information sent during the initial setup of the communication is large because N puzzles, consequently N keys, are sent initially.

1.2 Diffie-Hellman (1976)

The work by Diffie and Hellman[1] proposed a method such that only one "key" needs to be exchanged and in addition the time required from Jade to perform cryptanalysis is exponential. In addition, it allows authentication because its use allows it to be tied to a public file of user information. Shiela can authenticate Froi and vice versa.

Diffie and Hellman differentiate *public key cryptosystems* and *public key distribution systems*. We let K be the finite key space from which keys K can be obtained and M be the finite message space where messages M are derived. A *public key cryptosystem* is a pair of families of algorithms E_k andD_k which represent invertible transformations[1].

$$E_k : \{M\} \rightarrow \{M\}$$

$$D_k : \{M\} \rightarrow \{M\}$$

such that

1. for every key K, E_k is the inverse of D_k,
2. for every K and M, the algorithms E_k and D_k are easy to compute,
3. for almost every K, each easily computed algorithm equivalent to D_k is computationally infeasible to derive from E_k,
4. for every K, it is feasible to compute inverse pairs E_k and D_k from K.

Property 3 allows E_k to be made public without compromising D_k. Key distribution in this system is simplified. Users generate two keys, an enciphering key E and a deciphering key D. E can be made public but D is kept privately by the user. Any entity who would like to send messages to a user can use the publicly available E to encrypt messages but only the user can decrypt the message using D. In their paper, Diffie and Hellman gave an example public key cryptosystem by multiplying a binary n-vector message m with an invertible binary n x n matrix E. However, this approach is not practical.

Merkle's[4] work was classified by Diffie and Hellman as *public key distribution system* and highlighted its limitations specifically its high transmission overhead again because of sending N puzzles initially. The proposed system is similar to the public key cryptosystem described above, but unlike Merkle's technique, Diffie and Hellman approach allows the authentication of users by making the public file read-only[1].

The technique proposed is dependent on the difficulty of computing $logsmodq$ where **q** is a prime number representing the number of elements of a finite field. Users generate independent random numbers X_i from the set of integers $\{1, 2, ..., q-1\}$. The users keep these numbers but the computed value

$$Y_i = \alpha^{X_i} mod\, q$$

is placed publicly together with the user information(such as name and email).

Consider for example Shiela and Froi would like to talk to each other privately. They are going to use the key $K_{Shiela,Froi}$below after they generate $X_{Shiela}$and X_{Froi} and published Y_{Shiela} and Y_{Froi}.

$$K_{Shiela,Froi} = \alpha^{X_{Shiela}X_{Froi}} mod\, q$$

Shiela will be able to obtain $K_{Shiela,Froi}$ by using the public file Y_{Froi} and then computing

$$K_{Shiela,Froi} = Y_{Froi}^{X_{Shiela}} mod\, q$$

$$= (\alpha^{X_{Froi}})^{X_{Shiela}} mod\, q$$

$$= \alpha^{X_{Froi}X_{Shiela}} = \alpha^{X_{Shiela}X_{Froi}} mod\, q$$

Froi will be able to obtain the key in the same manner.

$$K_{Sheila,Froi} = Y_{Shiela}^{X_{Froi}} mod\, q$$

Jade might be able to compute$K_{Shiela,Froi}$ from Y_{Shiela} and Y_{Froi} by computing

$$K_{Shiela,Froi} = Y_{Shiela}^{(\log_\alpha Y_{Froi})} mod\, q$$

However, if Jade is to perform this computation, it will take him a long time to do so. This system takes advantage of the fact that $logs\, mod\, q$ are expensive to compute.

1.3 Rivest-Shamir-Adleman (1978)

The RSA algorithm by Rivest, Shamir, and Andleman was inspired by the work of Diffie and Hellman. Despite the breakthrough in Diffie and Hellman's work in public key cryptosystems, they did not present any practical implementation that can be used in actual systems. The creators of RSA took the work further by presenting a practical and efficient implementation. Given an encryption procedure **E** ,a decryption procedure **D**, and a message **P**, a public key cryptosystem has the following properties[5]:

1. Decrypting an encrypted **P** results to **P**. **D**(**E**(**P**)) = **P.**
2. **D** and **E** are easy to compute.
3. Publicly revealing **E** does not mean that it will be easy to compute **D** from **E**. It should be difficult or inefficient to compute **D** from **E**.
4. If **P** is decrypted and then encrypted, **P** is the result. **E**(**D**(**P**)) = **P**.

The encryption and decryption functions rely on a key such that the security of the functions or procedures rests on the security of the key. If **E** satisfies properties 1-3 is referred to as "trap-door one way function". If it also satisfies property 4 then it is referred to as "trap-door one-way permutation". In public key cryptosystems the usual "setup" time is simply the time it takes to make the encryption function public.[1] If Shiela wants to encrypt a message **P**, for Froi, with the key **(e, n)**, she must first represent **P** as an integer between **0** and **n-1**, where **e** and **n** are positive integers. **P** is then encrypted to generate the ciphertext **C** by raising **P** to the e^{th} power modulo **n**. The decryption process will raise the ciphertext to another power **d** modulo **n**. The mathematical formulation is shown below.

$$C \equiv E(P) \equiv P^e (mod\, n)$$

$$P \equiv D(C) \equiv C^d (mod\, n)$$

The length of the original message is not increased during the encryption process. The pairs **(e,n)** and **(d,n)** are the encryption keys and decryption keys respectively. The encryption key is made public and the decryption key is private to the user. In the above example, **(e,n)** is the public key of Froi, he can decrypt the message using **(d,n)** which is in his possession. The security of the approach is based on the security of the keys, thus the keys must be selected well. **n** is computed as a product of two large random primes **p** and **q**, **n** = **p** * **q**. Although **n** will be made public, **p** and **q** will not, which hides the way **d** can be derived from **e**. **d** is a random integer such that $gcd(d, (p-1)*(q-1)) = 1$. **e** is computed from **p**, **q**, and **d** such that

[1] In their paper, it is the encryption function that is made public. Other researchers talk about keys being made public, not the functions. Essentially, however, there is a one-to-one correspondence between the encryption function and the key.

$$e * d \equiv 1(mod(p-1) * (q-1))$$

This requirement guarantees that properties 1 and 4 above are satisfied, which means that **E** and **D** are inverse permutations. Although Diffie and Hellman[1] also used exponentiation and modulo in determining a common key to be used, their approach is not based on a "trap-door one-way permutation" which means that property 4 is satisfied.

The creators of RSA presented an efficient implementation of the of their approach. Note that the basic operations in the encryption and decryption process are exponentiation and division (to compute the remainder). The technique is called "exponentiation by repeated squaring and multiplication." The technique presented will need at most $2 * \log_2(e)$ multiplications and $2 * \log_2(e)$ divisions. The encryption and decryption algorithms are shown below.

```
1  long E(long P,int e, int n)
2  {
3      long C=1;
4      int i;
5
6      for (i=k;k==0;k--)
7      {
8          C = (C * C) % n;
9          if (bit(i,e) == 1) /* bit(i,e) returns the ith bit of e*/
10             C = (C * P) % n;
11     }
12     return C;
13 }
14
15 long D(long C,int d, int n)
16 {
17     long P=1;
18     int i;
19
20     for (i=k;k==0;k--)
21     {
22         P = (P * P) % n;
23         if (bit(i,d) == 1) /* bit(i,d) returns the ith bit of d*/
24             P = (P * C) % n;
25     }
26     return P;
27 }
```

It can be seen that the RSA encryption and decryption algorithms are straightforward to implement. The challenge however is in the selection of the keys which is dependent of several parameters (**e**, **d**, **n**). In their paper,

the creators of RSA also presented several approaches how to generate these parameters to guarantee the security.

1.4 Elgamal (1985)

The RSA[5] system depends on the difficulty of factoring large integers. Elgamal's work is based on the difficulty of computing logarithms over finite fields[2]. This approach is similar to Diffie and Hellman but is more practical for implementation. In Diffie and Hellman, **q** is a prime number representing the number of elements of a finite field (we will use **p** instead for consistency with Elgamal's paper). In order for make it difficult to compute the discrete logarithm, **p** must be chosen so that **p-1** have at least one large prime factor. Suppose Shiela wants to send a message **m** to Froi where $0 \leq m \leq (p-1)$. Shiela chooses a number **k** uniformly between **0** and **p-1**. This **k** will be X_{Shiela}, which is her secret key. $K_{Shiela,Froi}$ is computed as follow:

$$K_{Shiela,Froi} = Y_{Froi}^{k} mod\, p$$

Y_{Froi} is public. Then the ciphertext **C** that will be generated is the pair (c_1, c_2), such that

$$c_1 = \alpha^{k} mod\, p$$

$$c_2 = K_{Shiela,Froi} m\ mod\, p$$

Since the ciphertext is a pair(c_1, c_2), then its size is twice the size of the message **m**. When Froi receives **C**, the process of decrypting is divided into two parts. First is to recover $K_{Shiela,Froi}$using $c_1^{X_{Froi}} mod\, p$. Note that X_{Froi} is Froi's secret key which only he knows. The second step is to divide c_2 by $K_{Shiela,Froi}$ which recover the message **m.**

$$m = \frac{c_2}{K_{Shiela,Froi}}$$

The randomization in the encryption process (selection of **k**) is unique to Elgamal's method. This results to the ciphertext **C** of the message **m** not being repeated. This provide added security because Jade will not be able to obtain a pattern for running the encryption process repeatedly in case he guesses that the message is indeed **m**. Another notable difference is that the structure of the system does not disclose obvious relations between two messages m_1 and m_2 and their simple functions such as the concatenation $m_1 m_2$[2] .

1.5 Elliptic Curve Cryptosystems (1987)

The cryptosystems described above is based on the multiplicative group of a finite field $GF(q), q = p^n$[3]. An elliptic curve E_k define over a field K of characteristic $\neq 2$ or 3 is the set of solutions $(x, y) \in K^2)$ to the equation

$$y^2 = X^2 + ax + b, \quad a, b \in K$$

The main impact of elliptic curve cryptography is enhanced security because the counterpart of the discrete logarithms problem (which is the basis of the previously discussed methods) in elliptic curves is harder[3].

References

[1] Whitfield Diffie and Martin Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[2] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469–472, July 1985.

[3] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[4] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[5] Ronald L. Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.