Computer Projects Designed to Enhance Student's Learning Experience with Public-Key Cryptography

Abstract

Cryptography plays a fundamental role in safeguarding today's information infrastructure. Public-key cryptography is a cryptographic approach utilized by many cryptographic algorithms and cryptosystems. In contrast to symmetric key systems, it eliminates the need to share a key secretly. This distinguishing characteristic makes it a widely and successfully used technology around the world. It is the foundation for public-key infrastructure (PKI) and Internet standards such as Transport Layer Security (TLS) and Pretty Good Privacy (PGP). A thorough understanding of public-key cryptography is indispensable to not only engineering and science students, but also engineering technology students in the general fields of computing, networking, communications and information technology.

This paper describes an approach to teaching public-key cryptography to electrical and computer engineering technology students utilizing three computer projects designed to provide hands-on experience with public-key cryptography. These projects introduce students JAVA BigInteger class and its built-in methods and open source cryptography libraries such as crypto++ allowing students develop public-key cryptographic applications. Instead of using a small modulus for solely instructional demonstration, these projects allow student's natural curiosity to be stimulated and result in a deeper understanding of real world applications. To date, feedback from students has been very positive.

Introduction

With the increasing dependence of industry, businesses, education and society on computing and digital communications, the need for providing security through effective and efficient cryptographic algorithms has became more important than ever. Cryptography is the science of using mathematics to encrypt and decrypt data. Besides its traditional role of ensuring confidentiality, it has been utilized to ensure integrity, authentication, and non-repudiation which are the basic requirements in today's information systems or data communications. It is imperative to teach cryptography to students in the general areas of computing, information, networking and data communications. Recently, educators have also confirmed the importance of teaching encryption basics to general students [1].

Public-key cryptography is one of the major topics in our computer security course. Thought students seem to be very interested in this topic, teaching public-key cryptography is somewhat challenging since understanding the theory requires a high level of mathematical knowledge and skills. This particularly presents a challenge to engineering technology students. This paper shares our experience of teaching engineering technology students public-key cryptography. The paper is organized as follows. First, it briefly introduces the public-key cryptography basics and describes our approach to teach public-key cryptography. Then, it describes the computer projects we developed to enhance the student's learning experience. Finally, it illustrates the

sample projects accomplished by students in our computer security class taught last year, and presents our conclusion.

Basic Concepts and Teaching Approach

Prior to teaching public-key cryptography, the authors introduce basic security requirements within the context of application to application communications over the Internet. Four security requirements concepts of confidentiality, integrity, authentication and non-repudiation are introduced to students. In addition, concrete examples are used so that students are aware that each ensures one aspect of information security---authentication is the process of confirming or establishing something (or someone) as authentic, confidentiality ensures privacy so that no one else except the intended receiver can read the message. Integrity ensures the receiver that the received message has not been altered in any way from the original and non-repudiation is a mechanism to prove that the sender really sent this message. Upon the completion of this learning module, students should be able to identify and comprehend these requirements.

We start to teach students cryptography with a traditional cryptography (i.e., one-time pad, Caesar cipher and Wheatstone-Playfair cipher[2] etc.,) where both the sender and receiver of a message know and use the same secret key; the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is also known as secret key or symmetric cryptography. Students are guided to discuss how to use cryptographic schemes to achieve the security requirements mentioned above and identify the problem of how to communicate the secret key in an open environment such as Internet applications and E-business. A packet sniffer (i.e., Wireshark[3]) is utilized to demonstrate that confidentiality can be compromised if messages are exchanged without encryption. This naturally leads to the discussion of an asymmetric cryptography and its applications. The theory of public-key algorithms is beyond the scope the course. Consequently, only a brief outline of the operation of public-key algorithms will be given. The emphasis is to show students that it works, and involve them in developing public-key cryptosystem applications.

As a relatively new cryptographic approach, public-key cryptography's distinguishing characteristic is the use of a pair of keys including a secret private key and a published public-key which, unlike the symmetric key algorithms, does not require a secure initial exchange of secret key between the sender and receiver. The RSA cryptosystem, named after its inventors R. Rivest, A. Shamir, and L. Adleman, is the most widely used public-key cryptosystem. RSA cryptosystem is used as an example to teach public-key cryptography.

As illustrated in Figure 1, public-key algorithms use a pair of keys. One key is used for encryption and the other is used for decryption. The keys are chosen so that if one is used to encrypt a message the other must be used to decrypt and vice versa. They are chosen in such a way that even if an attacker knows one of them, finding the other is computationally infeasible due to the intractability of the integer factorization problem. The general idea of ensuring confidentiality is to use a public-key, which can be made available to the public or distributed in an open environment, to encrypt the message, and the cipher text can only be decrypted by the corresponding private key.

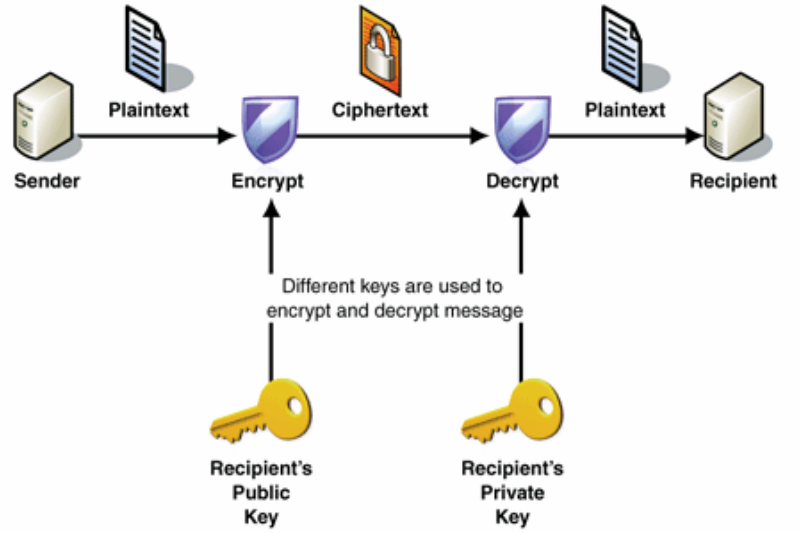

Figure 1: Illustration of Public-key Encryption and Decryption[4]

Students have always been curious to this feature. At this point, we use an instructional example, as listed in Table 1, to involve students into a public-key cryptosystem including key generation, encryption and decryption.

Table 1: Public-key Algorithm

Step	Description	Example
Step 1	Randomly select two prime numbers, denoted by *P* and *Q*	i.e., P=11, Q=17
Step 2	Compute the modulus *M=P*Q*, *M* is made publicly available	*M*=11*17=187
Step 3	Compute the Euler totient as *T= (P-1)* (Q-1)*	*T*=(11-1)*(17-1)=160
Step 4	Randomly select a public-key, *e*, such that (1) *e* <*T*; (2) *e* and *T* are relative prime numbers	*e* =13
Step 5	Find a "matched" private key *d* such that $ed-1$ can be evenly divided by *T*	*d* =37 (13*37-1=480, 480/160=3)
Step 6	To encrypt a plaintext, *N*, raise it to the *e*th power modulo *M*	$c=10^{13} \bmod 187=164$
Step 7	To decrypt a cipher text, *c*, raise *c* to the *d*th power modulo *M*	$N=164^{37} \bmod 187=10$
Step 8	What happens if the same key is used to decrypt the cipher text?	$N'=164^{13} \bmod 187=109$
Step 9	Encryption with private key	$7^{37} \bmod 187=28$
Step 10	Decryption with public key	$28^{13} \bmod 187=7$
Step 11	What happens if the same key is used to decrypt the cipher text?	$28^{37} \bmod 187=129$

Steps 1 through 5 outline the key generation process. Steps 6 through 8 illustrate encryption process with public-key and decryption process with private-key or public-key. Steps 9 through 11 demonstrate encryption with private key and decryption with public-key or private key.

For encryption process, the cipher text, c, is obtained by $N^e \bmod M$ where N is the numerical plaintext and mod denotes modulo operation, while for decryption process, the plaintext is recovered by $c^d \bmod M$. These exercises show students how RSA works and verify that ONLY the other key can be used to decrypt the cipher text.

Computer Projects to Enhance Student's Learning Experience

Three computer project assignments have been created to enhance student's learning experience. The first one is to have students understand which key to use for confidentiality and digital signature, respectively. The second project is to implement a public-key cryptosystem with modulus length of 768 or 1024 bits. The third project is to developed public-key cryptosystem applications using open source library crypto++.

Project 1: Which Key to Use?

Once students complete the exercise illustrated in Table 1. We teach them the principle of digital signature[4], the electrical world's counterpart to a handwriting signature. It always occurs to us that students need exercises to gain hands-on experience with which key and whose key should be used for confidentiality or digital signature. Pendegraft[5] has developed an inexpensive device to show it to students and confirmed that it is a common confusion to students at this point. In our laboratory class, the instructor creates a private key, public-key for a student upon his or her request, or students can generate their own public-keys and private keys by themselves. Each student's public-key is published on the blackboard. A letter is converted to an integer number by its alphabet order. Each student needs to team with at least another one to practice using his or her private key and his or her partner's public-key to achieve confidentiality or perform digital signature.

By using the partner's public-key, each student practices encrypting and sending cipher text to the partner. Meanwhile, each student also practices decrypting the received cipher text using his own private key. Every cipher text with the key used for encryption is published on the blackboard.

The rest of this project is to let every student simply encrypt his or her own name using his or her own private key. This exercise basically shows them a simplified (without hashing) "digital signature" technique. By performing this exercise, students understand authentication, integrity and non-repudiation can be achieved by using digital signature, and understand the logic behind digital signature---- the private key is secretly kept by its owner only, and the signature is verifiable since the matched public-key can be published.

It may take a while for students to be skillful enough to pick up the right key (whose key and which key) for confidentiality or digital signature. Once they have practiced a few rounds, it seems that they all understand the concepts quite well.

Project 2: Implementing a 768 or 1024-Bit Public-Key Cryptosystem

This computer project requires students develop a JAVA program for a 768 or 1024- bit modulus public-key cryptosystem. They are required to:

(1) Implement the key generation algorithm, i.e., steps 1 through 5 as illustrated in Table 1;
(2) Accept a plain message from the key board and print it out;
(3) Convert the message into integer numbers at the student's choice and print the numbers out;
(4) Encrypt the numbers from (3) using the public-key generated in (1) and print the result out;
(5) Decrypt the cipher text from (4) using the private key and print the decrypted message out;
(6) Compare results from (5) and (2).

Without using any existing resources, it will be very challenging for engineering technology students to complete this weekly project. However, Java offers a class named BigInteger[6] which provides immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations. JAVA BigInteger class together with its built-in methods makes the implementation of RSA public-key algorithms fairly straightforward and fast. The following examples and demos are introduced to students so that they can utilize these resources to complete the project.

To generate a random prime number p, it simply needs to add the following codes

```
SecureRandom randomNumber = new SecureRandom();
BigInteger p = new BigInteger(modulusbits / 2, 100, randomNumber)
```

where modulusbits are the number of bits for the modulus. It can be either 768 or 1024, depending on the student's choice.

BigInteger class has built-in methods including add, subtract, multiply and divide. Given the two random number p, q and theModulus, the calculation of the modulus can be done by:

```
theModulus = p.multiply(q)
```

and the Euler Totient can be calculated by

```
EulerTotient= p.subtract(p.ONE).multiply(q.subtract(q.ONE))
```

A public-key can be randomly generated by the method introduced above and the following conditions have to be tested prior to acceptance:

```
theModulus.compareTo(publicKey) == 1
publicKey.gcd(EulerTotient)==1
```

Note that the first statement ensures the public-key selected is less than the modulus and the second statement ensures the selected public-key and Euler Totient are relatively prime numbers.

Once a public-key has been found, the next step is to find the corresponding private key which can be obtained by the following code:

```
privateKey = publicKey.modInverse(EulerTotient)
```

which basically finds a BigInteger privateKey such that (privateKey*publicKey) mod (EulerTotient) = 1.

Most of our students have basic JAVA programming experience, once they are familiar with the statements above; they are capable to complete the programming assignment. A sample work is shown in next section.

Project 3: Application of Crypto++

Crypto++ Library[7] authored by Wei Dai (http://weidai.com), is a free C++ class library of cryptographic schemes. Currently the library contains most of the symmetric cryptographic algorithms (i.e., AES and other block ciphers) and asymmetric cryptographic schemes such as RSA, DSA and key exchange protocols[7]. The library is a powerful and elegant tool for performing complex cryptography. It uses advanced C++ features such as templates, multiple inheritance, and exceptions to achieve that power and elegance. For people who are familiar with C++, the library will appear intuitive and easy to use. Others may need to view it as a learning opportunity.

Sufficient information regarding how to use the library is available online. There are four sources of documentation for Crypto++. They are the source code, the Crypto++ Usenet group, the Crypto++ FAQ, and the Crypto++ Wiki[7-8]. In addition, a user guide and help file authored by Dennis Bider is also available[9].

In the laboratory class, students are guided to use these recourses. Students need to download the open source code from cryptopp.com, and build a static library and the instructors show them how to incorporate the library into the Microsoft Visual C++ (MSVC) integrated development environment (IDE) [8]. To integrate crypto++ library into MSVC IDE, the compiled library should be moved to the location of the header and source files, and then the location of the header files, source files, and libraries should be added to the VC++ Environment, and finally the location of the header files, source files, and libraries should be added to MSVC Project.

The sample project (called Cryptest) provided with the source code package demonstrates both symmetric and public-key library functions. After reviewing those demonstrations, students are asked to modify the source codes and build a digital signature application including key generation process, encryption, signing a document and then verify the document.

Student's Project Samples

We observed that students were very interested in these exercises. These three projects ranging from concept demonstration to practical application development meet the diverse needs of students and have them gain hands-on experience with mathematically challenging cryptography. This experience is helpful to the later study of public-key infrastructure topic as well.

Instead of using small modulus for solely instructional demonstration, the second project allows student's natural curiosity to be stimulated resulting in a deeper understanding of real world applications. Since the instructors have provided the necessary information of BigInteger and its build-in methods and how to use those methods for public-key cryptography, students are able to develop public-key cryptosystem applications. Figure 2 is one of the projects completed by students in the computer security class. In addition to implementing key generation, encryption and decryption, the students have developed a GUI together with RSA cracking demonstration.

Figure 2: A Student's Project

To date, feedbacks from students regarding computer projects 1 and 2 have been very positive. Meanwhile, we found that a few students struggled with project 3. This may be due to the lack of sufficient C++ programming skills or the lack of necessary detailed examples of library

functions. Our future work focuses on developing a good set of sample programs using crypto++ library so that students can modify from those example programs and extend to new applications.

Conclusion

Today, public-key cryptography is indispensable to ensure both confidentiality and integrity in numerous communication, networking and Internet applications. A thorough understanding of public-key cryptography is essential for engineering and technology professionals. Public-key cryptography such as RSA is mathematically challenging to engineering technology students, the concepts and its application can be taught via carefully designed computer projects. Three computer exercises have been developed to enhance student's learning experience. Feedback from students, teaching evaluation and student's learning outcomes show its effectiveness.

Bibliography

1. A. Temkin, "Teaching Cryptography to Continuing Education Students", IFIP International Federation for Information Processing. Vol. 237. Fifth World Conference on Information Security Education 2007.
2. Matth Bishop, Introduction to Computer Security, Addison-Wesley, 2005.
3. WireShark, Online resource, http://www.wireshark.org.
4. Online resource, Public-key Encryption and Digital Signatures, http://msdn.microsoft.com/en-us/library/aa480610.aspx.
5. N. Pendegraft , "An Inexpensive Device for Teaching Public-key Encryption", Journal of Information Systems Education, Vol. 20, No.3, 2009.
6. Java BigInteger Class, http://java.sun.com/j2se/1.4.2/docs/api/java/math/BigInteger.html.
7. Online resource at http://www.cryptopp.com.
8. G. Lancaster, J. Walton, "Compiling and Integrating Crypto++ into the Microsoft Visual C++ Environment", http://www.codeproject.com/KB/tips/CryptoPPIntegration.aspx
9. Crypto++ User Guide, http://www.bitvise.com/users-guide.html.