DIFF88 Diffie, W. "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988.

SHAM03 Shamir, A., and Tromer, E. "On the Cost of Factoring RSA-1024." *CryptoBytes*, Summer 2003. http://www.rsasecurity.com/rsalabs

KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Terms

chosen ciphertext attack (CCA)	private key	time complexity
digital signature	public key	timing attack
key exchange	public-key cryptography	trap-door one-way function
one-way function	public-key cryptosystems	
optimal asymmetric encryption padding (OAEP)	public-key encryption	
	RSA	

Review Questions

What are the principal elements of a public-key cryptosystem?

What are the roles of the public and private key?

What are three broad categories of applications of public-key cryptosystems?

What requirements must a public-key cryptosystems fulfill to be a secure algorithm?9.5

What is a one-way function?

What is a trap-door one-way function?

Describe in general terms an efficient procedure for picking a prime number.

Problems

Prior to the discovery of any specific public-key schemes, such as RSA, an existence proof was developed whose purpose was to demonstrate that public-key encryption is possible in theory. Consider the functions $f_1(x_1) = z_1$; $f_2(x_2, y_2) = z_2$; $f_3(x_3, y_3) = z_3$, where all values are integers with $1 \leq x_i, y_i, z_i \leq N$. Function f_1 can be represented by a vector M1 of length N, in which the kth entry is the value of $f_1(k)$. Similarly, f_2 and f_3 can be represented by $N \times N$ matrices M2 and M3. The intent is to represent the encryption/decryption process by table lookups for tables with very large values of N. Such tables would be impractically huge but could be constructed in principle. The scheme works as follows: Construct M1 with a random permutation of all integers between 1 and N; that is, each integer appears exactly once in M1. Construct M2 so that each row contains a random permutation of the first N integers. Finally, fill in M3 to satisfy the following condition:

$$f_3(f_2(f_1(k), p), k) = p \qquad \text{for all } k, p \text{ with } 1 \leq k, p \leq N$$

To summarize,

1. M1 takes an input k and produces an output x.
2. M2 takes inputs x and p giving output z.
3. M3 takes inputs z and k and produces p.

The three tables, once constructed, are made public.

a. It should be clear that it is possible to construct M3 to satisfy the preceding condition. As an example, fill in M3 for the following simple case:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

Convention: The ith element of M1 corresponds to $k = i$. The ith row of M2 corresponds to $x = i$; the jth column of M2 corresponds to $p = j$. The ith row of M3 corresponds to $z = i$; the jth column of M3 corresponds to $k = j$.

b. Describe the use of this set of tables to perform encryption and decryption between two users.

c. Argue that this is a secure scheme.

2 Perform encryption and decryption using the RSA algorithm, as in Figure 9.5, for the following:

a. $p = 3; q = 11, e = 7; M = 5$
b. $p = 5; q = 11, e = 3; M = 9$
c. $p = 7; q = 11, e = 17; M = 8$
d. $p = 11; q = 13, e = 11; M = 7$
e. $p = 17; q = 31, e = 7; M = 2$

Hint: Decryption is not as hard as you think; use some finesse.

3 In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $e = 5, n = 35$. What is the plaintext M?

4 In an RSA system, the public key of a given user is $e = 31$, $n = 3599$. What is the private key of this user? *Hint:* First use trial-and-error to determine p and q; then use the extended Euclidean algorithm to find the multiplicative inverse of 31 modulo $\phi(n)$.

5 In using the RSA algorithm, if a small number of repeated encodings give back the plaintext, what is the likely cause?

6 Suppose we have a set of blocks encoded with the RSA algorithm and we don't have the private key. Assume $n = pq$, e is the public key. Suppose also someone tells us they know one of the plaintext blocks has a common factor with n. Does this help us in any way?

7 In the RSA public-key encryption scheme, each user has a public key, e, and a private key, d. Suppose Bob leaks his private key. Rather than generating a new modulus, he decides to generate a new public and a new private key. Is this safe?

8 Suppose Bob uses the RSA cryptosystem with a very large modulus n for which the factorization cannot be found in a reasonable amount of time. Suppose Alice sends a message to Bob by representing each alphabetic character as an integer between 0 and 25 ($A \rightarrow 0, \ldots, Z \rightarrow 25$) and then encrypting each number separately using RSA with large e and large n. Is this method secure? If not, describe the most efficient attack against this encryption method.

9 Using a spreadsheet (such as Excel) or a calculator, perform the operations described below. Document results of all intermediate modular multiplications. Determine a number of modular multiplications per each major transformation (such as encryption, decryption, primality testing, etc.).

a. Test all odd numbers in the range from 233 to 241 for primality using the Miller-Rabin test with base 2.
b. Encrypt the message block $M = 2$ using RSA with the following parameters: $e = 23$ and $n = 233 \times 241$.
c. Compute a private key (d, p, q) corresponding to the given above public key (e, n).

d. Perform the decryption of the obtained ciphertext
   1. without using the Chinese Remainder Theorem, and
   2. using the Chinese Remainder Theorem.

10 Assume that you generate an authenticated and encrypted message by first applying the RSA transformation determined by your private key, and then enciphering the message using recipient's public key (note that you do NOT use hash function before the first transformation). Will this scheme work correctly [i.e., give the possibility to reconstruct the original message at the recipient's side, for all possible relations between the sender's modulus n_S and the recipient's modulus n_R ($n_S > n_R$, $n_S < n_R$, $n_S = n_R$)]? Explain your answer. In case your answer is "no," how would you correct this scheme?

11 "I want to tell you, Holmes," Dr. Watson's voice was enthusiastic, "that your recent activities in network security have increased my interest in cryptography. And just yesterday I found a way to make one-time pad encryption practical."

"Oh, really?" Holmes' face lost its sleepy look.

"Yes, Holmes. The idea is quite simple. For a given one-way function F, I generate a long pseudorandom sequence of elements by applying F to some standard sequence of arguments. The cryptanalyst is assumed to know F and the general nature of the sequence, which may be as simple as S, S + 1, S + 2, … , but not secret S. And due to the one-way nature of F, no one is able to extract S given F(S + i) for some i, thus even if he somehow obtains a certain segment of the sequence, he will not be able to determine the rest."

"I am afraid, Watson, that your proposal isn't without flaws and at least it needs some additional conditions to be satisfied by F. Let's consider, for instance, the RSA encryption function, that is $F(M) = M^K \bmod N$, K is secret. This function is believed to be one-way, but I wouldn't recommend its use, for example, on the sequence $M = 2, 3, 4, 5, 6, \ldots$"

"But why, Holmes?" Dr. Watson apparently didn't understand. "Why do you think that the resulting sequence $2^K \bmod N$, $3^K \bmod N$, $4^K \bmod N, \ldots$ is not appropriate for one-time pad encryption if K is kept secret?"

"Because it is—at least partially—predictable, dear Watson, even if K is kept secret. You have said that the cryptanalyst is assumed to know F and the general nature of the sequence. Now let's assume that he will obtain somehow a short segment of the output sequence. In crypto circles, this assumption is generally considered to be a viable one. And for this output sequence, knowledge of just the first two elements will allow him to predict quite a lot of the next elements of the sequence, even if not all of them, thus this sequence can't be considered to be cryptographically strong. And with the knowledge of a longer segment he could predict even more of the next elements of the sequence. Look, knowing the general nature of the sequence and its first two elements $2^K \bmod N$ and $3^K \bmod N$, you can easily compute its following elements."

Show how this can be done.

12 Show how RSA can be represented by matrices M1, M2, and M3 of Problem 9.1.

13 Consider the following scheme:
   1. Pick an odd number, E.
   2. Pick two prime numbers, P and Q, where $(P - 1)(Q - 1) - 1$ is evenly divisible by E.
   3. Multiply P and Q to get N.
   4. Calculate $D = \dfrac{(P - 1)(Q - 1)(E - 1) + 1}{E}$

Is this scheme equivalent to RSA? Show why or why not.

14 Consider the following scheme by which B encrypts a message for A.
   1. A chooses two large primes P and Q that are also relatively prime to $(P - 1)$ and $(Q - 1)$.
   2. A publishes $N = PQ$ as its public key.

3. A calculates P' and Q' such that $PP' \equiv 1 \pmod{Q-1}$ and $QQ' \equiv 1 \pmod{P-1}$.
4. B encrypts message M as $C = M^N \bmod N$.
5. A finds M by solving $M \equiv C^{P'} \pmod{Q}$ and $M \equiv C^{Q'} \pmod{P}$.

a. Explain how this scheme works.
b. How does it differ from RSA?
c. Is there any particular advantage to RSA compared to this scheme?
d. Show how this scheme can be represented by matrices M1, M2, and M3 of Problem 9.1.

15 "This is a very interesting case, Watson," Holmes said. "The young man loves a girl, and she loves him too. However, her father is a strange fellow who insists that his would-be son-in-law must design a simple and secure protocol for an appropriate public-key cryptosystem he could use in his company's computer network. The young man came up with the following protocol for communication between two parties. For example, user A wishing to send message M to user B: (messages exchanged are in the format sender's name, text, receiver's name)"

1. A sends B the following block: $(A, \mathrm{E}(PU_b, [M, A]), B)$.
2. B acknowledges receipt by sending to A the following block: $(B, \mathrm{E}(PU_a, [M, B]), A)$.

"You can see that the protocol is really simple. But the girl's father claims that the young man has not satisfied his call for a simple protocol, because the proposal contains a certain redundancy and can be further simplified to the following:"

1. A sends B the block: $(A, \mathrm{E}(PU_b, M), B)$.
2. B acknowledges receipt by sending to A the block: $(B, \mathrm{E}(PU_a, M), A)$.

"On the basis of that, the girl's father refuses to allow his daughter to marry the young man, thus making them both unhappy. The young man was just here to ask me for help."

"Hmm, I don't see how you can help him." Watson was visibly unhappy with the idea that the sympathetic young man has to lose his love.

"Well, I think I could help. You know, Watson, redundancy is sometimes good to ensure the security of protocol. Thus, the simplification the girl's father has proposed could make the new protocol vulnerable to an attack the original protocol was able to resist," mused Holmes. "Yes, it is so, Watson. Look, all an adversary needs is to be one of the users of the network and to be able to intercept messages exchanged between A and B. Being a user of the network, he has his own public encryption key and is able to send his own messages to A or to B and to receive theirs. With the help of the simplified protocol, he could then obtain message M user A has previously sent to B using the following procedure:"

Complete the description.

16 Use the fast exponentiation algorithm of Figure 9.8 to determine $5^{596} \bmod 1234$. Show the steps involved in the computation.

17 Here is another realization of the fast exponentiation algorithm. Demonstrate that it is equivalent to the one in Figure 9.8.

```
1. f ← 1; T ← a; E ← b
2. if odd(e) then f ← f × T
3. E ← [ E/2 ]
4. T ← T × T
5. if E > 0 then goto 2
6. output f
```

18 The problem illustrates a simple application of the chosen ciphertext attack. Bob intercepts a ciphertext C intended for Alice and encrypted with Alice's public key e. Bob wants to obtain the original message $M = C^d \bmod n$. Bob chooses a random value r less than n and computes

$$Z = r^e \bmod n$$
$$X = ZC \bmod n$$
$$t = r^{-1} \bmod n$$

Next, Bob gets Alice to authenticate (sign) X with her private key (as in Figure 3), thereby decrypting X. Alice returns $Y = X^d \bmod n$. Show how Bob can use the information now available to him to determine M.

19 Show the OAEP decoding operation used for decryption that corresponds to the encoding operation of Figure 10.

20 Improve on algorithm P1 in Appendix 9A.

a. Develop an algorithm that requires $2n$ multiplications and $n + 1$ additions. *Hint*: $x^{i+1} = x^i \times x$.

b. Develop an algorithm that requires only $n + 1$ multiplications and $n + 1$ additions. *Hint:* $\mathrm{P}(x) = a_0 + x \times q(x)$, where $q(x)$ is a polynomial of degree $(n - 1)$.

Note: The remaining problems concern the knapsack public-key algorithm described in Appendix J.

21 What items are in the knapsack in Figure F.1?

22 Perform encryption and decryption using the knapsack algorithm for the following:

a. $\mathbf{a}' = (1, 3, 5, 10); w = 7; m = 20; \mathbf{x} = 1101$

b. $\mathbf{a}' = (1, 3, 5, 11, 23, 46, 136, 263); w = 203; m = 491; \mathbf{x} = 11101000$

c. $\mathbf{a}' = (2, 3, 6, 12, 25); w = 46; m = 53; \mathbf{x} = 11101$

d. $\mathbf{a}' = (15, 92, 108, 279, 563, 1172, 2243, 4468); w = 2393; m = 9291; \mathbf{x} = 10110001$

9.23 Why is it a requirement that $m > \sum_{1=1}^{n} a'_i$?

APPENDIX 9A THE COMPLEXITY OF ALGORITHMS

The central issue in assessing the resistance of an encryption algorithm to cryptanalysis is the amount of time that a given type of attack will take. Typically, one cannot be sure that one has found the most efficient attack algorithm. The most that one can say is that, for a particular algorithm, the level of effort for an attack is of a particular order of magnitude. One can then compare that order of magnitude to the speed of current or predicted processors to determine the level of security of a particular algorithm.

A common measure of the efficiency of an algorithm is its time complexity. We define the **time complexity** of an algorithm to be $\mathrm{f}(n)$ if, for all n and all inputs of length n, the execution of the algorithm takes at most $\mathrm{f}(n)$ steps. Thus, for a given size of input and a given processor speed, the time complexity is an upper bound on the execution time.

There are several ambiguities here. First, the definition of a step is not precise. A step could be a single operation of a Turing machine, a single processor machine instruction, a single high-level language machine instruction, and so on. However, these various definitions of step should all be related by simple multiplicative constants. For very large values of n, these constants are not important. What is important is how fast the relative execution time is growing. For example, if we are concerned about whether to use 50-digit ($n = 10^{50}$) or 100-digit ($n = 10^{100}$) keys for RSA, it is not necessary (or really possible) to know exactly how long it would take to break each size of key. Rather, we are interested in ballpark figures for level of effort and in knowing how much extra relative effort is required for the larger key size.

A second issue is that, generally speaking, we cannot pin down an exact formula for $\mathrm{f}(n)$. We can only approximate it. But again, we are primarily interested in the rate of change of $\mathrm{f}(n)$ as n becomes very large.

There is a standard mathematical notation, known as the "big-O" notation, for characterizing the time complexity of algorithms that is useful in this context. The definition is as follows: $f(n) = O(g(n))$ if and only if there exist two numbers a and M such that

$$|f(n)| \leq a \times |g(n)|, \quad n \geq M \tag{9.3}$$

An example helps clarify the use of this notation. Suppose we wish to evaluate a general polynomial of the form

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

The following simple algorithm is from [POHL81].

```
algorithm P1;
    n, i, j: integer; x, polyval: real;
    a, S: array [0..100] of real;
    begin
       read(x, n);
       for i := 0 upto n do
       begin
           S[i] := 1; read(a[i]);
           for j := 1 upto i do S[i] := x × S[i];
           S[i] := a[i] × S[i]
       end;
       polyval := 0;
       for i := 0 upto n do polyval := polyval + S[i];
       write ('value at', x, 'is', polyval)
    end.
```

In this algorithm, each subexpression is evaluated separately. Each S[i] requires ($i + 1$) multiplications: i multiplications to compute S[i] and one to multiply by a[i]. Computing all n terms requires

$$\sum_{i=0}^{n} (i + 1) = \frac{(n + 2)(n + 1)}{2}$$

multiplications. There are also ($n + 1$) additions, which we can ignore relative to the much larger number of multiplications. Thus, the time complexity of this algorithm is $f(n) = (n + 2)(n + 1)/2$. We now show that $f(n) = O(n^2)$. From the definition of Equation (9.3), we want to show that for $a = 1$ and $M = 4$ the relationship holds for $g(n) = n^2$. We do this by induction on n. The relationship holds for $n = 4$ because $(4 + 2)(4 + 1)/2 = 15 < 4^2 = 16$. Now assume that it holds for all values of n up to k [i.e., $(k + 2)(k + 1)/2 < k^2$]. Then, with $n = k + 1$,

$$\begin{aligned}
\frac{(n + 2)(n + 1)}{2} &= \frac{(k + 3)(k + 2)}{2} \\
&= \frac{(k + 2)(k + 1)}{2} + k + 2 \\
&\leq k^2 + k + 2 \\
&\leq k^2 + 2k + 1 = (k + 1)^2 = n^2
\end{aligned}$$

Therefore, the result is true for $n = k + 1$.

In general, the big-O notation makes use of the term that grows the fastest. For example,

1. $O[ax^7 + 3x^3 + \sin(x)] = O(ax^7) = O(x^7)$
2. $O(e^n + an^{10}) = O(e^n)$
3. $O(n! + n^{50}) = O(n!)$

There is much more to the big-O notation, with fascinating ramifications. For the interested reader, two of the best accounts are in [GRAH94] and [KNUT97].

An algorithm with an input of size n is said to be

- **Linear:** If the running time is $O(n)$
- **Polynomial:** If the running time is $O(n^t)$ for some constant t
- **Exponential:** If the running time is $O(t^{h(n)})$ for some constant t and polynomial $h(n)$

Generally, a problem that can be solved in polynomial time is considered feasible, whereas anything worse than polynomial time, especially exponential time, is considered infeasible. But you must be careful with these terms. First, if the size of the input is small enough, even very complex algorithms become feasible. Suppose, for example, that you have a system that can execute 10^{12} operations per unit time. Table 9.6 shows the size of input that can be handled in one time unit for algorithms of various complexities. For algorithms of exponential or factorial time, only very small inputs can be accommodated.

The second thing to be careful about is the way in which the input is characterized. For example, the complexity of cryptanalysis of an encryption algorithm can be characterized equally well in terms of the number of possible keys or the length of the key. For the Advanced Encryption Standard (AES), for example, the number of possible keys is 2^{128}, and the length of the key is 128 bits. If we consider a single encryption to be a "step" and the number of possible keys to be $N = 2^n$, then the time complexity of the algorithm is linear in terms of the number of keys $[O(N)]$ but exponential in terms of the length of the key $[O(2^n)]$.

Table 6. Level of Effort for Various Levels of Complexity

Complexity	Size	Operations
$\log_2 n$	$2^{10^{12}} = 10^{3\times 10^{11}}$	10^{12}
N	10^{12}	10^{12}
n^2	10^6	10^{12}
n^6	10^2	10^{12}
2^n	39	10^{12}
$n!$	15	10^{12}