Appendix J
The Knapsack Algorithm

William Stallings

Supplement to
Cryptography and Network Security, Sixth Edition
Prentice Hall 2013
ISBN: 0133354695
http://williamstallings.com/Cryptography

A number of algorithms have been proposed for public-key cryptography. Some of these, though initially promising, turned out to be breakable. It is instructive to review the most important such scheme.

J.1 THE KNAPSACK PROBLEM

The most famous of the fallen contenders is the trapdoor knapsack proposed by Ralph Merkle [MERK78]. The knapsack problem deals with determining which objects are in a container, such as a knapsack. A simple example is shown if Figure J.1. The knapsack is filled with a subset of the items shown, whose weights in grams are indicated. Given the weight of the filled knapsack, 1156 grams, the problem is to determine which of the items are contained in the knapsack. (The scale is calibrated to deduct the weight of the empty knapsack.) As an exercise, the reader is encouraged to determine the contents of the knapsack by trail-and-error calculation.

The problem illustrated in Figure J.1 is relatively simple but generally becomes computationally formidable when there are, say, 100 items rather than the 10 of this example. Merkle's contribution was to show (1) how to turn the knapsack problem into a scheme for encryption and decryption, and (2) how to incorporate trapdoor information that would enable a person to quickly solve the knapsack problem.

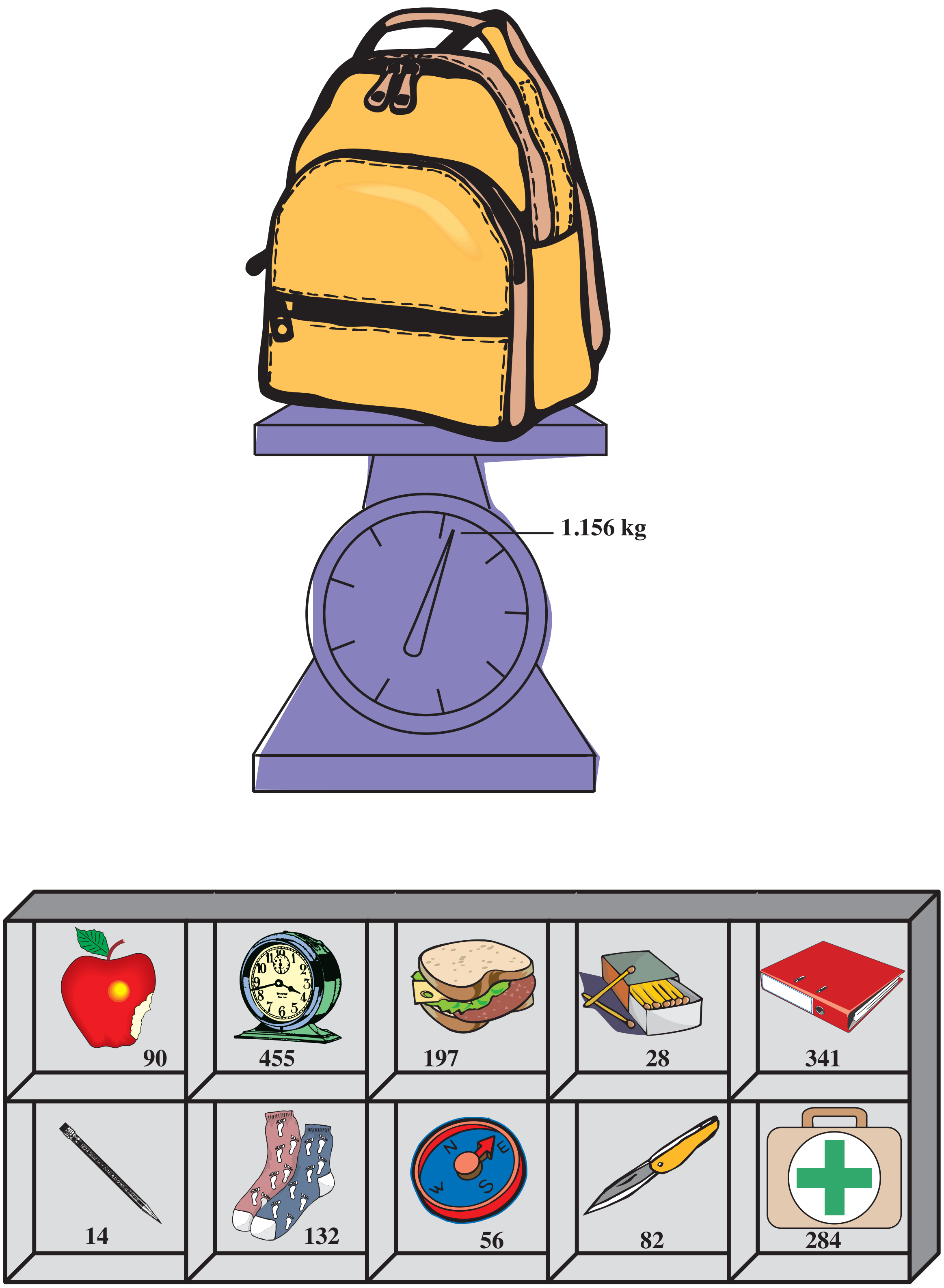

Figure J.1 Illustration of the Knapsack Problem

J.2 THE KNAPSACK CRYPTOSYSTEM

First, let us state the general approach for encryption/decryption using the knapsack problem. Suppose we wish to send messages in blocks of *n* bits. Then, define:

cargo vector	$\mathbf{a} = (a_1, a_2, \ldots, a_n)$	a_i integer
plaintext message block	$\mathbf{x} = (x_1, x_2, \ldots, x_n)$	x_i binary
corresponding ciphertext	$S = \mathrm{a} \bullet \mathrm{x} = \sum_{i=1}^{n} \left(a_i \times x_i\right)$	

Consider the cargo vector **a** to be a list of potential elements to be put in the knapsack, with each vector element in **a** equal to the weight of the corresponding cargo element. And consider the plaintext message block **x** to be a selection of elements from the cargo vector, with $x_i = 1$ for each cargo element a_i that is selected for inclusion in the knapsack. Thus each unique plaintext message block corresponds to a unique selection of items from the cargo vector. The vector product *S* is simply the sum of the selected items, which is the weight of the knapsack.

For encryption, **a** is the public key. If Bob wishes to send a confidential message **x** to Alice, Bob encrypts the message using Alice's public key **a**. Bob performs $S = \mathbf{a} \bullet \mathbf{x}$ and transmits *S*. For decryption, the Alice must recover **x**, given *S* and **a**.

This public-key scheme must satisfy two requirements. The **first requirement** is that there be a unique inverse for each value of *S*. For example, consider the following:

$$\mathbf{a} = (1, 3, 2, 5)$$
$$S = 3$$

This problem has two solutions: **x** = 1010 and **x** = 0100. Thus, the elements of **a** must be chosen such that each combination of elements yields a unique value.

The **second requirement** is that decryption is hard in general but easy if special knowledge is available. (In the preceding example, the special knowledge would function as Alice's private key.) Certainly, for large values of *n*, the knapsack problem is hard in general. But, under special circumstances, the problem is easy to solve. Suppose we impose the condition that each element of **a** is larger than the sum of the preceding elements:

$$a_i > \sum_{j=1}^{i-1} a_j \quad 1 < i \le n \tag{J.1}$$

This is known as a *superincreasing vector*. In this case, the solution is easy. For example, consider the vector

$$\mathbf{a'} = (171, 197, 459, 1191, 2410)$$

which satisfies condition (J.1). Suppose we have $S' = \mathbf{a'} \bullet \mathbf{x'} = 3798$. Because 3798 > 2410, a_5 must be included ($x_5 = 1$), because without a_5 the other elements cannot contribute enough to add up to 3798. Now consider

3798 – 2410 = 1388. Because 1388 > 1191, a_4 must also be included (x_4 = 1). Continuing in this fashion, we find that $x_3 = 0$, $x_2 = 1$, and $x_1 = 0$. Thus, in this example, given the public key **a'** and the encrypted message *S'*, it is possible to decrypt the message without access to a private key.

What Merkle did was find a way to tie an easy superincreasing knapsack problem to a difficult general knapsack problem. Suppose we choose at random an easy superincreasing knapsack vector $\mathrm{a}' = \left(a_1', a_2', \ldots a_n'\right)$, with *n* elements. Also select two integers *m* and *w*, such that *m* is greater than the sum of the elements of **a'** and *w* is relatively prime to *m*. That is:

$$m > \sum_{1=1}^{n} a_i'$$

$$\gcd(w, m) = 1$$

Now, we construct a hard knapsack vector **a** by multiplying the easy vector **a'** by *w*, modulo *m*:

$$\mathbf{a} = w\mathbf{a}' \bmod m$$

The vector **a** will, in general, not be superincreasing and can therefore be used to construct hard knapsack problems. However, knowledge of *w* and *m* enables the conversions of this hard knapsack problem to an easy one. To see this, first observe that because *w* and *m* are relatively prime, there exists a unique multiplicative inverse w^{-1}, modulo *m*. Therefore,

$$w^{-1}\mathbf{a} \equiv \mathbf{a}' \pmod{m}$$

We are now ready to define the knapsack scheme. The ingredients are the following:

a', a superincreasing vector	(private, chosen)
m, an integer larger than $\sum_{1=1}^{n} a'_i$	(private, chosen)
w, an integer relatively prime to m	(private, chosen)
w^{-1}, the inverse of w, modulo m	(private, calculated)
a, equal to w**a'**mod m	(public, calculated)

The private key consists of the triple (w^{-1}, m, **a'**). The public key is **a**. Suppose that Alice has published her public key **a** and that Bob wishes to send the message **x** to Alice. Bob calculates the sum:

$$S = \mathbf{a} \bullet \mathbf{x}$$

The determination of **x** given S and **a** is difficult, so this is a secure transmission. On receipt, Alice is able to decrypt easily. Define $S' = w^{-1}S \bmod m$. We have:

$$S = \mathbf{a} \bullet \mathbf{x} = w\mathbf{a'} \bullet \mathbf{x}$$
$$S' = w^{-1}S \bmod m$$
$$S' = w^{-1}w\mathbf{a'} \bullet \mathbf{x} \bmod m$$
$$S' = \mathbf{a'} \bullet \mathbf{x}$$

Therefore, we have converted the hard problem of finding **x** given S and **a** to the easy problem of finding **x** given S' and **a'**.

The knapsack algorithm was hailed as an unbreakable system. Merkle, confident though not rich, offered a reward of $100 to anyone who could break it. It took four years, but Adi Shamir, one of the inventors of RSA, broke the system and collected the $100 [SHAM82].

But Merkle was not through. He observed that the hard knapsack problem could be made even harder by using multiple transformations (w_1, m_1), (w_2, m_2), and so on. The overall transformation that results is not equivalent to any single (w, m) transformation. With this in mind, Merkle upped the ante to $1000 for anyone who could break the multiple-iteration problem. This time he had only two years to wait before having to pay up [ADLE83]. This ended serious consideration of knapsacks as a basis for public-key cryptography.

J.3 EXAMPLE

Figure J.2 shows an example. Alice creates a private key by first generating a superincreasing vector **a'** = (1, 3, 7, 13, 26, 65, 119, 267). Then, Alice selects an integer greater than $\Sigma a_i = 501$; the prime $m = 523$ is selected. The advantage of choosing a prime number for m is that all positive integers less than m are relatively prime to m. Alice chooses $w = 467$. Alice then computes the inverse of w modulo 523, which is $w^{-1} = 28$; that is, $(467 \times 28) \bmod 523 = 1$. To complete the public key, Alice calculates $\mathbf{a} = w\mathbf{a'} \bmod m$ = (467, 355, 131, 318, 113, 21, 135, 215).

Bob now has Alice's public can and can encrypt messages to Alice. Given the plaintext message $\mathbf{x}$ = 01001011, Bob computes $S = \mathbf{a} \bullet \mathbf{x} = 818$. To decrypt the ciphertext, Alice first computes $S' = w^{-1}S \bmod m = (28 \times 818) \bmod 523 = 415$, and then solves the easy knapsack problem to recover 01001011.

Key Generation

easy knapsack **a'**	1	3	7	13	26	65	119	267

modulus m = 523 multiplier w = 467 w^{-1} = 28

$(1 \times 467)_{\text{mod } 523} = 467$	$(26 \times 467)_{\text{mod } 523} = 113$
$(3 \times 467)_{\text{mod } 523} = 355$	$(65 \times 467)_{\text{mod } 523} = 21$
$(7 \times 467)_{\text{mod } 523} = 131$	$(119 \times 467)_{\text{mod } 523} = 135$
$(13 \times 467)_{\text{mod } 523} = 318$	$(267 \times 467)_{\text{mod } 523} = 215$

hard knapsack **a**	467	355	131	318	113	21	135	215

public key PU_A = **a** private key $PR_A = (w^{-1}, m, \mathbf{a'})$

Encryption

Plaintext = 01001011
Ciphertext = $(0 \times 467) + (1 \times 355) + (0 \times 131) + (0 \times 318) + (1 \times 113) + (0 \times 21) + (1 \times 135) + (1 \times 215)$
= 818

Decryption

$(818 \times w^{-1})_{\text{mod } m} = (28 \times 818) \text{ mod } 523 = 415$

$415 \geq 267 \Rightarrow a_8 = 1$	$29 - 26 = 3 < 13 \Rightarrow a_4 = 0$
$415 - 267 = 148 \geq 119 \Rightarrow a_7 = 1$	$3 < 7 \Rightarrow a_3 = 0$
$148 - 119 = 29 < 65 \Rightarrow a_6 = 0$	$3 \geq 3 \Rightarrow a_2 = 1$
$29 \geq 26 \Rightarrow a_5 = 1$	$3 - 3 = 0 < 1 \Rightarrow a_1 = 0$

Plaintext = $a_8a_7a_6a_5a_4a_3a_2a_1$ = 01001011

Figure J.2 Knapsack Example

J.4 REFERENCES

ADLE83 Adleman, L. "The Function Field Sieve." *Lecture Notes in Computer Science 877, Springer-Verlag*, 1983

MERK78 Merkle, R., and Hellman, M. "Hiding Information and Signatures in Trap Door Knapsacks." *IEEE Transactions on Information Theory*, September 1978.

SHAM82 Shamir, A. "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem." *Proceedings, 23rd IEEE Symposium on the Foundations of Computer Science*, 1982.