Merkle-Hellman Knapsack Cryptosystem in Undergraduate Computer Science Curriculum

Y. Kortsarts[1], **Y. Kempner**[2]
[1]Computer Science Department, Widener University, Chester, PA, USA
[2]Computer Science Department, Holon Institute of Technology, Holon, Israel

Abstract *- We present our experience integrating Merkle-Hellman knapsack cryptosystem into undergraduate computer science curriculum. The paper focuses on the additive knapsack and ways to integrate it into the following undergraduate computer science courses: introduction to computer science, cryptology, and analysis of algorithms. Ideas for undergraduate student projects are presented and discussed.*

Keywords: Merkle-Hellman knapsack cryptosystem, undergraduate computer science curriculum

1 Introduction

In classical symmetric or private-key cryptosystems the encryption and decryption keys are either the same or can be easily found from each other. A new type of cryptosystem, call a public-key cryptosystem was introduced in 1976 by Diffie and Hellman [2]. In public-key cryptosystem, different keys are used for encryption and decryption, and the fact that one knows how to encrypt the message does not mean that it can be easily decrypted. The first public-key cryptosystem, RSA, was developed by Ronald L. Rivest, Adi Shamir and Leonard Adleman [3] and was based on integer factorization. R.C. Merkle and M.E. Hellman created the first knapsack public-key cryptosystem in 1978. The Merkle-Hellman cryptosystem is based on the subset sum problem, a special case of the knapsack problem. The general knapsack problem is an NP-complete combinatorial problem that is known to be computationally difficult to solve in general.

The history has not been kind to knapsack schemes [11]. Merkle offered $100 award for breaking singly - iterated knapsack and in 1982 Adi Shamir developed a polynomial time algorithm for breaking the basic Merkle - Hellman cryptosystem [4,5,6] using Hendrik W. Lenstra's algorithm [7] for the integer programming problem when the number of variables is fixed. At the CRYPTO '83 conference, Adleman used an Apple II computer to demonstrate Shamir's method [8]. Later, Merkle offered $1000 award for breaking multiply iterated knapsack and it was broken by Brickell in 1985 [9], a system of 40 iterations was breaking in about an hour of Cray-1 time. The development of the new public-key cryptosystems based on knapsack problem continued well into 1990s [8] and there are some modern research results in that area [15, 16]. There are many variants of knapsack cryptosystems and the history of their development and the history of the development of their cryptanalysis are very important.

2 Motivation

Integration Merkle-Hellman knapsack cryptosystem into computer science curriculum promotes development of mathematical reasoning. Mathematical reasoning is very critical in computing. It is the root for algorithm development, program validation and solution verification. There are deep connections between algorithmic and mathematical thinking. Devising efficient algorithms and demonstrating their practicality is an important application of mathematics in computer science and software engineering. Computer science students therefore need to exercise their mathematical as well as their computational abilities, and computer science educators need to help students use both ways of thinking to solve computing problems. It is important to promote the role of mathematics in computer science and software engineering education [17 – 24].

Merkle-Hellman knapsack cryptosystem provides an opportunity to make connections between mathematics and computer science topics due to elegant and rich underlying mathematical background. It is a real-world problem with the real-world applications, and it could be successfully "translated" to the undergraduate students' level. In addition, the problem is derived from the current computer science research and it helps to make connection between research and curriculum at the early stages of students' career.

3 Knapsack Cryptosystem in Computer Science Curriculum

Knapsack cryptosystem could be successfully integrated into various core and elective undergraduate computer science courses such as cryptology, analysis of algorithms and introduction to computer science. Due to its simple structure, the Merkle-Hellman knapsack cryptosystem is an ideal model for introducing public key cryptosystem in cryptology course. In design and analysis of algorithms course the problem could be related to general knapsack problem, subset-sum problem, various algorithmic techniques including dynamic

programming, and the concept of NP-completeness. In introductory programming courses, the knapsack cryptosystem is an excellent source for computational problems related to various number theory concepts such as prime numbers, great common divisor, Euclidian algorithm, modular exponentiation, and primitive roots for primes. This problem has a rich hands-on component and provides a possibility to design hands-on activities and interesting laboratory and homework assignments. Knapsack cryptosystem and its variants also provide interesting ideas for undergraduate student projects.

The paper mainly will discuss the additive knapsack – the simplest variant of the Merkle-Hellman knapsack cryptosystem. Multiplicative knapsack and multiply iterated knapsack will be briefly considered as well. We discuss integration of these knapsack cryptosystems into computer science courses and ideas for student projects in more detail later in the paper.

4 Theoretical Background

Consider general 0-1 knapsack problem: given n items of different values v_i and weights w_i, find the most valuable subset of the items while the overall weight does not exceed a given capacity W. The general knapsack problem is NP-hard [10], but could be solved in pseudo-polynomial time through dynamic programming algorithmic technique.

A subset sum problem is a special case of knapsack problem when a value of each item is equal to its weight. The input for subset sum problem is a set of positive integers: $A = \{a_1, a_2, \ldots, a_n\}$ and the positive integer S. If there is a subset of A that sums to S, the output is TRUE, and contains the subset itself, otherwise, the output is FALSE. The subset sum problem is NP-hard.

An easy knapsack problem is one in which set $A = \{a_1, a_2, \ldots, a_n\}$ is a super-increasing sequence. A super-increasing sequence is one in which the next term of the sequence is greater than the sum of all preceding terms:

$$a_2 > a_1, a_3 > a_1 + a_2, \ldots, a_n > a_1 + a_2 + \ldots + a_{n-1} \quad (1)$$

For example: $A = \{1, 2, 4, 8, \ldots, 2^{n-1}\}$ is a super-increasing sequence.

There is a polynomial time algorithm for easy knapsack problem. The input for the algorithm is a super-increasing sequence A and the value S. If there is a subset X of A that sums to S, the output of the algorithm is TRUE, and algorithm also outputs binary array P of n elements, where $P[i] = 1$ if and only if a_i, belongs to X. Otherwise, the algorithm returns FALSE.

Polynomial Time Algorithm:

```
for i ← n to 1
        if S ≥ a_i
    then P[i] ← 1 and S ← S-a_i
      else   P[i] ← 0
if S != 0
  then return (FALSE – no solution)
else return (P[1], P[2], …P[n]).
```

5 Merkle-Hellman Additive Knapsack Cryptosystem

We assume that Alice and Bob are two users that would like to communicate using additive knapsack cryptosystem [1] and we suppose that they communicate according to the following scheme:

Alice:

1. Chooses secret (*private*) key
2. Creates and publishes *public* key
3. Receives encrypted message - *ciphertext*
4. Decrypts ciphertext using secret key to recover the original message - *plaintext*

Bob

1. Uses public key to encrypt the plaintext
2. Sends ciphertext to Alice

In more detail, Alice is performing the following steps [1, 13]:

1. Chooses $A = \{a_1, a_2, \ldots, a_n\}$ super-increasing sequence – easy knapsack
2. Makes A to be her private key
3. Calculates $a_1 + \ldots + a_n = E$
4. Chooses $M > E$.
5. Chooses W that satisfies $2 \le W < M$ and $(W, M) = 1$, to ensure that W is invertible mod M
6. Computes Public (hard) knapsack $B = \{b_1, b, \ldots, b_n\}$, where $b_i = W a_i \pmod{M}$, $1 \le i \le n$
7. Keeps Private Key: *A, W, M*
8. Publishes Public key*: B*

Let us assume that Bob would like to encrypt plaintext P and to send ciphertext to Alice. Bob's encryption process goes through the following steps [1, 13]:

1. Binary Plaintext P breaks up into k sets each of n elements long: $P = \{P_1, \ldots, P_k\}$
2. For each set P_i compute
$$\sum_{j=1}^{n} P_{ij} b_j = C_i \quad (2)$$
3. C_i is the ciphertext that corresponds to plaintext P_i

4. $C = \{ C_1, \ldots, C_k)$ is ciphertext that corresponds to the plaintext P
5. C is sent to Alice

After receiving the ciphertext from Bob, Alice will perform the decryption process that consists of the following steps [1, 13]:

1. Since $(W, M) = 1$, W is invertible mod M and Alice computes w, the multiplicative inverse of W mod M:

$$wW \equiv 1 \pmod{M} \quad (3)$$

2. Alice uses connection between easy and hard knapsacks which defined as follows:

$$w\, b_i = a_i \pmod{M},\ 1 \le i \le n \quad (4)$$

3. For each C_i Alice computes:

$$S_i = w\, C_i \pmod{M} \quad (5)$$

$$S_i = wC_i(\mathrm{mod}\, M) = w\sum_{j=1}^{n} P_{ij} b_j (\mathrm{mod}\, M) =$$

$$= \sum_{j=1}^{n} P_{ij} wb_j (\mathrm{mod}\, M) = \sum_{j=1}^{n} P_{ij} a_j (\mathrm{mod}\, M) \quad (6)$$

4. Since $S_i < M$ and $M > E = a_1 + \ldots + a_n$, the final formula for finding plaintext would be

$$S_i = \sum_{j=1}^{n} P_{ij} a_j \quad (7)$$

5. Since, A is an easy knapsack, plaintext P_i could be found using polynomial time algorithm for easy knapsack that we described in the previous section.

To illustrate how additive knapsack cryptosystem works, we consider the following example (Figure 1) :

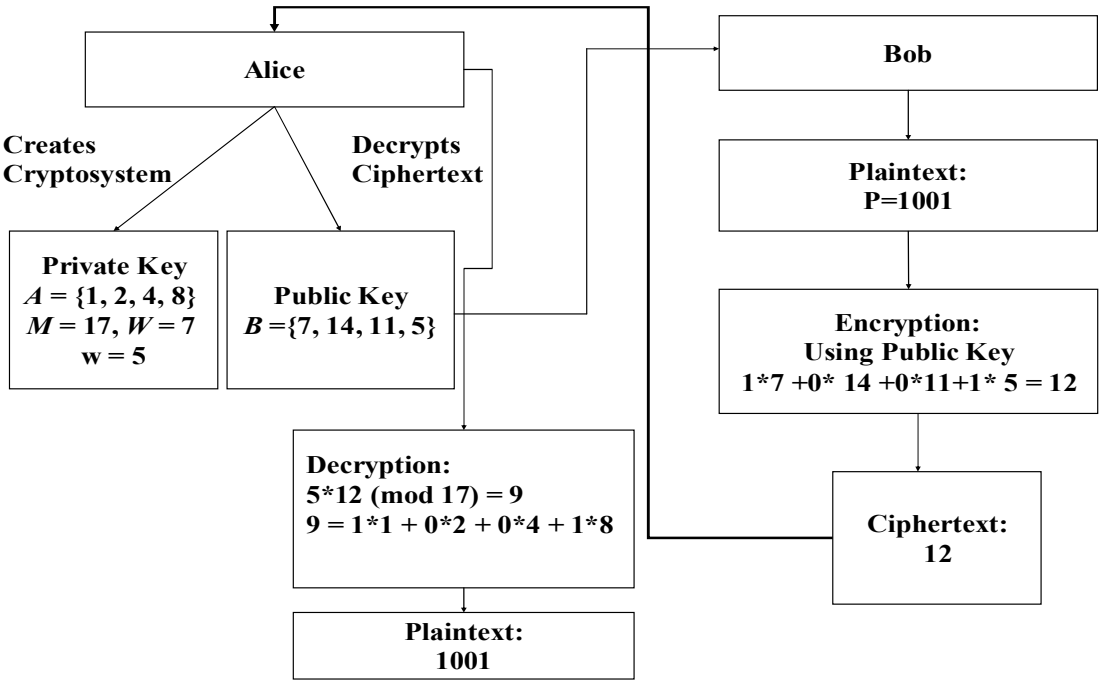

Figure 1 : Example

6 Dynamic Programming Algorithm

In this section, we explain the dynamic programming algorithm to break additive Merkle-Hellman knapsack. We assume that unauthorized person, an eavesdropper, knows only public key, $B=\{b_1, b_2, \ldots, b_n\}$ and ciphertext, C, and would like to recover the plaintext. This process known as *ciphertext-only cryptanalytic attack on Merkle-Hellman Knapsack*:

Input: $B = \{b_1, b_2, \ldots, b_n\}$ – public key, C - ciphertext
Output: The binary array P – plaintext
Algorithm: Let $Q[i, j]$ be TRUE if there is a subset of $b_1, b_2, \ldots, b_i$ that sums to j, $0 \le i \le n$, $0 \le j \le C$

Step 1: Computation of Q
$Q[0][0] \leftarrow$ TRUE
for $j = 1$ to C do: $Q[0][j] \leftarrow$ FALSE
for $i = 1$ to n do:
  for $j = 0$ to C do:
    if $(j - B[i] < 0)$: Q[i][j] = Q[i-1][j]
    else: Q[i][j] = Q[i-1][j-B[i]] *or* Q[i-1][j]

Step 2: Backtracking
Let P be an array of n elements initialized to 0
$i \leftarrow n, j \leftarrow C$
while $i > 0$:
  if $(j - B[i]) \ge 0)$:
    if ($Q[i$-1][j-$B[i]$] is True):
      $P[i] \leftarrow P[i] + 1$
      $j \leftarrow j - B$[i]

  $i \leftarrow i - 1$

Output: array P, elements of P that equal to 1 construct a desired subset of B that sums to C.
To illustrate the above algorithm, we will continue our example considered earlier:

Input: public key $B = \{7, 14, 11, 5\}$ and ciphertext $C = 12$

Below is the table that shows the values of $Q[i][j]$ for each i and j. Using the table and backtracking algorithm we are able to construct the plaintext vector P. In our example,
$P = \{1, 0, 0, 1\}$ which means that the desired plaintext is 1001.

Table 1: Dynamic Programming Example
Public key $B = \{7, 14, 11, 5\}$ and ciphertext $C = 12$

	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9	j=10	j=11	j=12
i = 0	T	F	F	F	F	F	F	F	F	F	F	F	F
i=1 B[1]=7	T	F	F	F	F	F	F	*T*	F	F	F	F	F
i=2 B[2]=14	T	F	F	F	F	F	F	T	F	F	F	F	F
i=3 B[3]=11	T	F	F	F	F	F	F	T	F	F	F	T	F
i=4 B[4]=5	T	F	F	F	F	T	F	T	F	F	F	T	*T*

7 Integrating Additive Knapsack into Computer Science Curriculum

As we mentioned earlier, various undergraduate courses would benefit from integration of additive knapsack cryptosystem into their curriculum.

First, we consider a cryptology course - the most natural choice. In this course, the additive knapsack would be introduced while covering public key cryptology concept. The additive knapsack cryptosystem is easy to comprehend and this cryptosystem is of historic interest. Taking in account the mathematical background of the students and the level of their preparation, simple variants of Merkle-Hellman knapsack – multiplicative and multiply iterated knapsacks [1] – could be integrated as well providing further enrichment of the course curriculum. Cryptology course has a minor disadvantage being

an elective course that is not always offered and not taken by all students.

In design and analysis of algorithms course, the additive knapsack is related to general knapsack problem, sub-set sum problem and dynamic programming algorithmic technique and will provide an interesting real-world example to illustrate the above-mentioned concepts. The lecture devoted to Merkle-Hellman knapsack could include a short introduction to cryptology providing definitions of the main concepts such as cryptology, cryptography, cryptanalysis, plaintext, ciphertext, encryption, decryption, and the encryption scheme (code). The cryptology topic is very well accepted by students and provides an opportunity to show connection between theoretical topics learned in class and practical applications. The introductory cryptology part could be followed by additive knapsack algorithm with examples, and the final part of the lecture could introduce the dynamic programming algorithm for ciphertext-only cryptanalysis. Similar to cryptology course, variants of additive knapsack could be considered as well. The knapsack cryptosystem topic will also enrich part of the course dealing with the NP-completeness connecting theoretical and practical issues.

In introductory programming courses, additive knapsack cryptosystem is an excellent source for programming assignments. This topic could be covered within one-two lecture hours and one three hours laboratory practice (this period is provided based on the schedule for introductory programming courses in our institution – 3 hours lectures and 3 hours laboratory weekly). The topic could be introduced after covering modular integer arithmetic, if-else statements, loops, functions, and one-dimensional and two-dimensional arrays – which are basic topics and usually covered in all introductory programming courses. As a preparation step, students would be able to write a computational solution for the following problems: (1) prime numbers checker; (2) relatively prime numbers checker; (3) computing the great common divisor using Euclidian algorithm; (4) computing the modular multiplicative inverse. The next step would be explaining basic cryptology concepts and additive knapsack encryption and decryption algorithms. After completing the theory explanation, students will be ready to write programming implementations of encryption and decryption. The programming assignment could consist of the following: (1) writing the function that generates super-increasing sequence; (2) using programs that were written at the preparation step, writing a function that computes the public knapsack; (3) writing a simple version of encryption function when the length of the binary plain text equals to the length of the public key – this will allow students to use only one dimensional arrays; (4) writing a simple decryption function; (5) writing a general encryption and decryption functions. As a simple example for super-increasing sequence, students could use the powers of 2. To culminate the explanation of the topic, the ciphertext-only cryptanalysis concept will be explained and the dynamic programming algorithm will be introduced, and students will be asked to write a programming implementation applying their knowledge of two-dimensional arrays.

Knapsack cryptosystem could be a topic for the interesting undergraduate research project or senior design project. If multiplicative and multiply iterated knapsacks are not covered in the regular courses, these Merkle-Hellman knapsack cryptosystem variants could be a good starting point for the project. For advanced projects, it is possible for students to research complicated variants of general knapsack cryptosystem [8, 13], design programming implementation of these systems, and create visualizations. The research of the cryptanalysis techniques could be considered as well. It would be also possible to create a project that implementing the polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem that was proposed by Adi Shamir [5, 6, 12, 14].

8 Summary

In this paper, we discussed the way to integrate basic Merkle-Hellman cryptosystem into undergraduate computer science curriculum. The topic provides unique opportunity to enrich various core and elective courses by connecting theory with practice, and promoting development of the mathematical reasoning.

9 Acknowledgment

This work was funded by Provost Grant at Widener University.

10 References

[1] R. C. Merkle, M. E. Hellman, Hiding Information and Signatures in Trapdoor Knapsacks, IEEE Transactions on Information Theory, vol. IT-24, pp. 525-530, 1978

[2] W. Diffie, M. E. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654, 1976

[3] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978

[4] A. Shamir. A Polynomial-time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. Proceedings of the IEEE Symposium on Foundations of Computer Science. IEEE, New York, pp. 145-152, 1982

[5] A. Shamir. A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. In David Chaum, Ronald L. Rivest, Alan T. Sherman. editors, Advances in Cryptology – CRYPTO '82. Plenum, New York, 1983.

[6] A. Shamir. A Polynomial-time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. IEEE Transactions on Information Theory, vol. IT-30, no. 5, September pp. 699-704, 1984

[7] H. W. Lenstra Jr, Integer Programming with a Fixed Number of Variables, Mathematics and Operations Research, vol. 8, no. 4, pp. 538-548, 1983

[8] M. K. Lai, Knapsack Cryptosystems: The Past and the Future, http://www.cecs.uci.edu/~mingl/knapsack.html

[9] E. F. Brickell, Breaking Iterated Knapsacks. In G. R. Blakley, David C. Chaum, editors, Advances in Cryptology – CRYPTO '84, Lecture Notes in Computer Science, vol. 196. Springer, Berlin, pp. 342-358, 1985

[10] M. Garey and D.S. Johnson, Computers and Intractability: A guide to the Theory of NP-Completeness, Freeman, 1979

[11] S. Goldwasser, M. Bellare, Lecture Notes on Cryptography, Summer course on cryptography, MIT, 1996-2001, http://cseweb.ucsd.edu/users/mihir/papers/gb.pdf

[12] J. C. Lagarias, Performance Analysis of Shamir's Attack on the Basic Merkle-Hellman Knapsack Cryptosystem. Proceedings of the 11th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 172. Springer, Berlin, 1984.

[13] A. M. Odlyzko. The Rise and Fall of Knapsack Cryptosystems. In Carl Pomerance, editor, Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics, vol. 42. American Mathematics Society, Providence, RI, pp. 75-88, 1990 http://www.dtc.umn.edu/~odlyzko/doc/complete.html

[14] A. M. Odlyzko. Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme. IEEE Transactions on Information Theory, IT-30, pp. 594-601,1984 http://www.dtc.umn.edu/~odlyzko/doc/complete.html

[15] A. M. Youssef, Cryptanalysis of a knapsack-based probabilistic encryption scheme, Information Sciences: an International Journal, v.179 n.18, p.3116-3121, August, 2009

[16] B. Wang, Q. Wu, and Y. Hu, A knapsack-based probabilistic encryption scheme, Information Sciences, vol. 177, no. 19, 3981-3994, 2007.

[17] D. Baldwin and P. Henderson, math-thinking discussion group web site http://www.math-in-cs.org/

[18] D. Knuth, The Art of Computer Programming, Volumes 1, 2, 3; Addison-Wesley.

[19] P. Henderson, Mathematical Reasoning in Software Engineering Education, September, Communications of the ACM, Volume 46, Issue 9, 2003

[20] D. Gries, The Science of Programming, Springer-Verlag, New York, 1981.

[21] D. Gries, and F. B. Schneider, A Logical Approach to Discrete Math, Springer-Verlag, New York, 1993.

[22] D. F. Butcher and W. A. Muth, "Predicting Performance in an Introductory Computer Science Course", Communications of the ACM, (28:3), pp. 263-268, 1985

[23] D. Ginat, "Early algorithm efficiency with design patterns", Computer Science Education, (11:1), pp. 1-21, 2000

[24] Henderson, P.B., Baldwin, D., et al, ''Striving for Mathematical Thinking,'' ITiCSE 2000 Working Group Report, SIGCSE Bulletin - Inroads, Vol. 33, No. 4, pp. 114-124, 2001