A tough knapsack: the Chor-Rivest Cryptosystem

Cameron Durham

MATH 453: Mathematical Cryptography

Professor: Dr. Gerard L. Ornas, McNeese State University

July 2018

Contents

1 Introduction 2

2 Knapsack Cryptosystems 2

3 Definitions and Theorems 4

4 The Cryptosystem 7

5 Attacks 11

6 Conclusion 12

References 13

7 $GF(125)$ 15

1 Introduction

From the 1980's to the 2000's, there were two main categories of public key cryptosystems [2]. The first was based on the computational hardness of number theoretic problems, such as factoring large composite numbers (as in the RSA) and computing discrete logarithms (like the El Gamal). These problems are widely regarded to be computationally intensive, even with a hidden trapdoor. The second category was based on the difficulty of the knapsack problem. These cryptosystems became popular due to their apparent speed improvement during decryption [9]. Of the proposed knapsacks, only the Chor-Rivest system withstood attack for a substantial period of time. In this paper, I will describe the Chor-Rivest cryptosystem, show a simple example, and discuss the security of the system. While the Chor-Rivest knapsack has been broken, the creativity of the system is are worthy of discussion.

2 Knapsack Cryptosystems

Suppose you are a traveler, going on an expedition with Maeve and Diego to the Valley Beyond. However, you can only bring one large knapsack. Given the knapsack volume and your list of belongings, how can you maximize its capacity for the long trip ahead? This is the knapsack problem, a.k.a. the subset-sum problem. If you have a list of items size k, the best knapsack algorithm requires $\approx 2^{k/2}$ operations to solve the problem [9] . This number quickly becomes impractical to compute as your knapsack size grows. Since the time taken by the algorithm grows exponentially, the problem is similar to the hard number-theoretic problems

of other public key cryptosystems. Also, a knapsack algorithm will produce a nonzero vector (the problem's solution) which could conceivably encode a message. Hence, the problem was attractive to cryptosystem designers. A formal definition of the problem follows:

The Knapsack Problem Given a set $\{v_i\}$ of k positive integers and an integer V, find a k-bit integer $A = (a_{k-1}, a_{k-2}, \cdots, a_1, a_0)$, $(a_i \in \{0, 1\}$ are the binary digits of $A)$ such that $\Sigma_{i=0}^{k-1} a_i v_i = V$, if such an A exists. [6]

Despite the hardness of the general knapsack problem, certain cases are easy to solve. Through a careful choice of the knapsack items V, one can make a sequence of weights appear difficult to solve. If the recipient has secret information (the private key) about V, they can solve the problem fairly quickly. Merkle and Hellman first proposed a modular arithmetic scheme to hide a superincreasing sequence [8]. Given a superincreasing sequence (that is, a set $\{v_i \mid v_{i+1} > \Sigma_{j=0}^{i} v_j \}$, where each element is greater than the sum of all preceding elements), we can easily solve the system. Since each consecutive element v_i is exponentially greater than the previous elements, we do not have to check it again to solve the reduced the knapsack problem. Thus, a superincreasing sequence could be recursively solved with this simple procedure:

1. Given a set of knapsack weights $V = \{v_0, \cdots, v_n\}$, find the largest element v_i less than the sum S.

2. Select and remove the item from the sum to solve the smaller knapsack problem with $n - 1$ elements

3. Repeat until the sum equals zero

Other cryptosystems proposed more complex schemes to hide low-density knapsacks [4]. The density of a knapsack refers to the ratio of the number of elements to the bitstring length of the largest element. Such sets are useful in knapsack cryptosystems since one could easily create a low-density sequence so that their subset sums are unique. As attractive as these sequences may sound, the underlying structure of the knapsacks made them easy to break.

The first knapsack systems used unique modular multiplication schemes to hide easy problems. Each eventually succumbed to attack through either cryptanalysis of the systems' modulus or lattice reduction techniques. In 1984, Brickell published a method to break the general knapsack cryptosystem without any knowledge of its construction [4]. Thus, Chor and Rivest sought to create a new type of knapsack using a high-density sequence that increased polynomially rather than exponentially.

3 Definitions and Theorems

Before describing the Chor-Rivest knapsack, I will provide a few definitions and theorems (from [5]) used throughout the cryptosystem.

Extension Field A field E is an extension of a field F if $F \subseteq E$ and the operations of F are those of E restricted to F

Classification of Finite Fields For each prime p and each positive integer h, there is, up to isomorphism, a unique finite field of order p^h.

Galois Field $GF(p^h)$ A finite field is called a Galois field of order p^h and is denoted by $GF(p^h)$

Subfields of a Finite Field For each divisor k of h, $GF(p^h)$ has a unique subfield of order p^k. Moreover, these are the only subfields of $GF(p^h)$.

Construction of $GF(p^h)$ The finite field $GF(p^h)$ is a field extension of $\mathbb{Z}_p$ of degree h. Each finite field $GF(p^h)$ can be constructed by finding a polynomial $f(x)$ of degree h, irreducible in $\mathbb{Z}_p[x]$, and defining

$$GF(p^h) = \mathbb{Z}_p[x]/(f(x))$$

To construct their cryptosystem, Chor and Rivest found the following theorem of Bose and Chowla useful [2] . We will let p be the sequence length and h be the desired subset size. The theorem shows the existence of a sequence $\{a_i | 1 \leq a_i \leq p^h - 1\}$ with the property that each sum of h elements (with repetition allowed) is unique. What follows is the Bose-Chowla theorem, slightly modified by Chor and Rivest to fit cryptographic applications:

Theorem 3.1 (Bose-Chowla) *Let p be prime, $h \geq 2$ an integer. Then there exists a sequence $A = \{a_i | 0 \leq i \leq p-1\}$ of integers such that:*

1. $1 \leq a_i \leq p^h - 1 (i = 0, 1, \ldots, p-1)$
2. *If $(x_0, x_1, \ldots, x_{p-1})$ and $(y_0, y_1, \ldots, y_{p-1})$ are two distinct vectors with non-negative integral coordinates and $\Sigma_{i=0}^{p-1} x_i, \Sigma_{i=0}^{p-1} y_i \leq h$, then $\Sigma_{i=0}^{p-1} x_i p_i \neq \Sigma_{i=0}^{p-1} y_i a_i$.*

[2], [1]

The proof of this theorem provides a method to construct such sequences. We will use the finite field $GF(p)$ as the base field and its h-degree extension $GF(p^h)$ (where p and h satisfy the the hypothesis of the theorem).

An Informal Proof:

We begin by choosing an element t from $GF(p^h)$ that is the root of the h-degree minimal polynomial over $GF(p)$. Let g generate the set $GF(p^h)^* = \{g^n | 0 \leq n \leq p^h - 1\}$. We now define each element of the set A to be the discrete logarithm of the base field shifted by t. That is, define a_i to be:

$$g^{a_i} = t + \alpha_i \quad \forall \alpha_i \in GF(p)$$

Now, we prove that each selection of at most h elements of A has a unique sum. Suppose there are two different selections of at most h items of A with the same sum. We can write these as vectors $\vec{x}, \vec{y}$, such that $\vec{x} \neq \vec{y}$, but $\Sigma_{i=0}^{p-1} a_i x_i = \Sigma_{i=0}^{p-1} a_i y_i$. Then we have the following equality:

$$g^{\Sigma_{i=0}^{p-1} a_i x_i} = g^{\Sigma_{i=0}^{p-1} a_i y_i}$$

However, we can write this as the product of the a_i powers of g:

$$\Pi_{i=0}^{p-1} (g^{a_i})^{x_i} = \Pi_{i=0}^{p-1} (g^{a_i})^{y_i}$$

$$(t + \alpha_0)^{x_0} \cdots (t + \alpha_{p-1})^{x_{p-1}} = (t + \alpha_0)^{y_0} \cdots (t + \alpha_{p-1})^{y_{p-1}}$$

Upon multiplying the last expression, each side becomes a monic h-degree polynomial. These polynomials are distinct, since of A elements distinct. Thus, subtraction yields a nonzero polynomial of degree $\leq h - 1$ over $GF(p)$ with t as its root, a contradiction. Therefore, each h-fold sum of A is unique.

This sketch of proof will help us construct the knapsack (our public key). We will generate a logarithm table, using element g, and define each element of A:

$$a_i = \log_g(t + \alpha_i)$$

Decryption involves raising g to our ciphertext $a_i \cdot x_i$ (our unique h-length sum) and then factoring the resulting polynomial to determine the indices i. We now will describe the cryptosystem in depth.

4 The Cryptosystem

We now give an description of how one would send and receive messages with the Chor-Rivest knapsack. Rigorous definitions are found in [2] and [7]. Suppose Maeve wishes to receive messages of length p from Diego. Her first step is to pick a prime power p and h such that $p^h - 1$ has no large prime factors. This will enable her to feasibly compute discrete logarithms in $GF(p^h)$. Next, Maeve picks an polynomial $f(t)$ (it should be a monic irreducible polynomial over $GF(p)$) of degree h. She will then represent the knapsack field as the set of polynomials degree $\leq h-1$ with coefficients in $\mathbb{Z}_p[t] = GF(p)[t]$ with all arithmetic done modulo p and $f(t)$. Then, Maeve picks a random generator g of $GF(p^h)$ and element $t \in GF(p^h)$. Following the Bose-Chowla Theorem, Maeve defines $a_i = \log_g(\alpha_i + t)$ for each α_i in the base field. These a_i's are scrambled with a random permutation π and a random integer $d < p^h - 1$ is chosen to add noise to each a_i. Meave sets $c_i = a_{\pi(i)} + d$ and publishes her public key: $\{(c_0, c_1, \ldots, c_{p-1}), p, h\}$. Her private key is: $\{t, g, \pi^{-1}, d\}$.

Encryption is a very easy task for Diego. A simple algorithm transforms his

message m into p length blocks with exactly h 1-bits. Then, Diego encrypts his message $M = (m_0 m_1 \dots m_{p-1})$ by simply sending the sum of the c_i elements corresponding 1-bit locations in his message:

$$E(M) = \Sigma_{i=0}^{p-1} m_i c_i \pmod{p^h - 1}.$$

Decryption is accomplished by Meave as follows. Upon receiving the message $E(M)$, she raises the generator to power of the sum of the knapsack elements and forms a polynomial of degree h over $GF(p)[t]$. The h roots of this polynomial are found by successive substitutions. When Maeve obtains the roots, she applies her inverse permutation to determine the indices of 1-bits in Diego's message.

Our discussion so far has been fairly vague. A toy example follows to illustrate the Chor-Rivest Knapsack in action.

Generating Public and Private Keys

1. Base Field: $GF(5) = \mathbb{Z}_5[t]$

2. Knapsack Extension Field: $GF(5^3) = \mathbb{Z}_5[t]/(f(t))$ where $f(t) = t^3 + 3t + 2$ is the degree $h = 3$ polynomial irreducible in the base field. The larger field will be constructed by adjoining the root of $f(t)$ to polynomials in $GF(5) = \mathbb{Z}_5[t]$.

3. Choosing a generator: Since t is a multiplicative generator of $GF(5^3)$, we raise t to a random integer $1 < r < 125 - 1$ and test if t^r is also a generator of $GF(5^3)$. In this manner, we select $g = t^{69} = 3t^2 + 3t + 3$

4. Computing logarithms: a logarithm table for $GF(5^3)$ is generated to compute

the following for each $\alpha_i \in GF(5)$. See section **7** ($GF(125)$ Table 2) for the complete table.

$$a_0 = \log_g(t+0) = \ \ 9$$
$$a_1 = \log_g(t+1) = \ 59$$
$$a_2 = \log_g(t+2) = \ 79$$
$$a_3 = \log_g(t+3) = \ \ 2$$
$$a_4 = \log_g(t+4) = \ 58$$

5. Disguising the knapsack: the randomly chosen integer ($d = 86$) and weights scrambled (using the permutation from Table 1) $c_i = a_{\pi(i)} + d \pmod{124}$

i	0	1	2	3	4
$\pi(i)$	1	0	3	2	4
$a_{\pi(i)}$	59	9	2	79	58

Table 1: $GF(p)$ permutation

6. Maeve uploads her public key and keeps her private key secure.

   Public Key: $\{C = (21, 95, 88, 41, 20),\ p = 5,\ h = 3\}$

   Private Key: $\{t,\ f(t) = 3t^2 + 3t + 3,\ \pi^{-1},\ d = 86\}$

Encryption

1. Our message will be $m = (22)_{10} \rightarrow (10110)_2$. The corresponding bit string is: $M = (m_0 m_1 m_2 m_3 m_4) = (10110)$.

2. Our encrypted message is:

$$E(M) = \Sigma_{j=0}^{p-1} c_j m_j \pmod{p^h - 1} = \Sigma_{j=0}^{4} c_j m_j \pmod{124} = 26$$

Decryption

1. Given ciphertext $E(M) = s$, remove the noise added during the system generation:

$$s' = s - h \cdot d \pmod{125} = 26 - 3 \cdot 86 \pmod{124} = 16$$

2. Raise the generator g to the sum s' (the sum of the scrambled discrete logarithms $a_{\pi(i)} \cdot m_i$) and define the $h - 1$ degree polynomial:

$$u(t) = g^{s'} \mod f(t) = (3t^2 + 3t + 3)^{16} = t^2 + 3t + 4$$

3. Define polynomial $s(t)$ to be the h degree polynomial in $GF(p)[t]$:

$$\begin{aligned} s(t) &= t^h + u(t) - (t^h \mod f(t)) \\ &= (t^2 + 3t + 4) + (t^3 + 3t + 2) \pmod{5} \\ &= t^3 + t^2 + t + 1 \end{aligned}$$

4. Factor the polynomial to recover the the $a_{\pi(i)}$ elements:

$$\begin{aligned} s(t) &= (t+1) \cdot (t+2) \cdot (t+3) \\ &= (t + a_{\pi(1)}) \cdot (t + a_{\pi(2)}) \cdot (t + a_{\pi(3)}) \end{aligned}$$

Since each $a_{\pi(i)}$ element reveals the positions of the $1's$ in the sender's message, we decrypt:

$$\pi^{-1}(1) = 0, \pi^{-1}(2) = 3, \pi^{-1}(3) = 2$$

Thus the decrypted message is:

$$m = (m_0 m_1 m_2 m_3 m_4) = (10110)_2 = (22)_{10}$$

5 Attacks

The following are a few brief examples of possible attacks on the Chor-Rivest knapsack. If the cryptanalyst knew elements of the private key, [2] shows how to crack the system in a reasonable amount of time. Their paper includes detailed descriptions of how to break the system if certain elements of the private key were known. When published, Brickell's attack was theoretically capable of breaking the system with no knowledge of the private key. The idea behind the attack was to find a small degree equation satisfied by the generator g through the construction of basis vectors for $GF(p^h)$. By solving a system of equations, one could construct an intricate equation that would produce t and allow reconstruction of the private key. However, even the most optimized version of this algorithm requires exhaustively searching the field $GF(p^h)$ for a solution. If a knapsack used the parameters proposed by Chor and Rivest, Brickell's attack would require $197^{8.8}$ operations [2] (note that this means with state-of-the-art 4GHZ chips, this operation could take over 1200 years to complete!). Hence, the Chor-Rivest knapsack was considered safe to use since keeping your private key secret is an expected for any respectable cryptosystem.

In 1998, Vaudenay [11] published an exhaustive attack on the system that provided a polynomial time algorithm to break the Chor-Rivest knapsack. The Vaudenay attack uses the fact that you can restructure certain types of Galois

fields by factoring h. An attacker could use a smaller subfield $GF(p^r)$ of $GF(p^h)$ to find a suitable generator. Since h/r is small, finding such a generator is feasible. Similar to Brickell's attack, this method yields t but in a reasonable amount of time (a few hours versus a thousand years). Hence, the private key could be obtained without too much trouble. Vaudenay proposed fixing the knapsack by requiring that h be prime and by hiding some of the actual knapsack elements, which he believed revealed too much information about the system.

6 Conclusion

In September 1978, Merkle and Hellman proposed the first knapsack cryptosystem [8]. Despite the apparent hardness of the knapsack problem, their scheme was quickly broken by Shamir in December of the same year [10]. Over the next few years, similar cryptosystems were proposed and promptly broken by exploiting flaws in the knapsack's structure. Chor and Rivest published their knapsack cryptosystem in 1988. By avoiding the failed low-density and superincreasing knapsacks, their scheme took longer to break. In 1998, Vaudenay published an attack that could crack the Chor-Rivest knapsack in polynomial time. Further research [3] revealed that if h was prime and $10^{44} < p^h - 1 < 10^{60}$, Vaudenay's attack would fail. However, discrete logarithms in a field this size would take days to compute on a standard PC. Thus, the Chor-Rivest Knapsack is unusable since it would either be insecure or infeasible given current computing power.

References

[1] R.C. Bose and S. Chowla, *Theorems in the additive theory of numbers*, Comment. Mth. Helvet. **37** (1962), 141–147.

[2] B. Chor and R.L. Rivest, *A knapsack type public key cryptosystem based on arithmetic in finite fields*, Advances in Cryptology: Proceedings of CRYPTO 84 (1985), 54–65.

[3] L. E. Hernandez, J. M. Masque, and D. Araceli, *Safer parameters for the chor–rivest cryptosystem*, **56** (2008), 2883–2886.

[4] Ernest F. Brickell and Andrew M. Odlyzko, *Cryptanalysis: A survey of recent results*, **76** (1988), 578 – 593.

[5] W.J. Gilbert, *Modern algebra with applications*, ch. Field Extensions, pp. 218–232, New York: John Wiley and Sons, 1976.

[6] N. Koblitz, *A course in number theory and cryptography*, ch. 4 Knapsack, pp. 111–115, Springer-Verlag, 1994.

[7] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of applied cryptography*, 1st ed., ch. Public-Key Encryption, pp. 302–306, CRC Press, Inc., Boca Raton, FL, USA, 1996.

[8] R. Merkle and M. Hellman, *Hiding information and signatures in trapdoor knapsacks*, IEEE Transactions on Information Theory **24** (1978), no. 5, 525–530.

[9] A. M. Odlyzko, *The rise and fall of knapsack cryptosystems*, In Cryptology and Computational Number Theory, A.M.S, 1990, pp. 75–88.

[10] A. Shamir and R. Zippel, *On the security of the merkle-hellman cryptographic scheme*, M.I.T. / Lab for Computer Science, Cambridge, Massachusetts 02139, December 1978.

[11] Serge Vaudenay, *Cryptanalysis of the chor-rivest cryptosystem*, Advances in Cryptology – CRYPTO '98 (Berlin, Heidelberg) (Hugo Krawczyk, ed.), Springer Berlin Heidelberg, 1998, pp. 243–256.

7 $GF(125)$

Computed using Sage with the generator $g = 3t^2 + 3t + 3$

g^1	$3t^2 + 3t + 3$	g^{33}	$3t + 4$	g^{65}	$3t^2 + 2t + 2$	g^{97}	$2t^2 + 2t + 3$
g^2	$t + 3$	g^{34}	$t^2 + 4t + 4$	g^{66}	$4t^2 + 4t + 1$	g^{98}	$3t^2 + 2t$
g^3	$2t^2 + 3t + 3$	g^{35}	$3t^2 + 3t + 2$	g^{67}	$t^2 + 4t$	g^{99}	$3t^2 + 3t$
g^4	$t^2 + t + 4$	g^{36}	$2t^2 + 3t$	g^{68}	$t^2 + t$	g^{100}	$t^2 + 2t + 4$
g^5	$4t^2 + t$	g^{37}	$2t^2 + 2t$	g^{69}	$2t^2 + 4t + 3$	g^{101}	$2t^2 + 4$
g^6	$4t^2 + 4t$	g^{38}	$4t^2 + 3t + 1$	g^{70}	$4t^2 + 3$	g^{102}	$2t$
g^7	$3t^2 + t + 2$	g^{39}	$3t^2 + 1$	g^{71}	$4t$	g^{103}	$t^2 + 3t + 3$
g^8	$t^2 + 2$	g^{40}	$3t$	g^{72}	$2t^2 + t + 1$	g^{104}	$2t^2 + t$
g^9	t	g^{41}	$4t^2 + 2t + 2$	g^{73}	$4t^2 + 2t$	g^{105}	$t^2 + 4t + 2$
g^{10}	$3t^2 + 4t + 4$	g^{42}	$3t^2 + 4t$	g^{74}	$2t^2 + 3t + 4$	g^{106}	$2t^2 + 2t + 1$
g^{11}	$t^2 + 3t$	g^{43}	$4t^2 + t + 3$	g^{75}	$4t^2 + 4t + 2$	g^{107}	$2t^2 + t + 4$
g^{12}	$3t^2 + 2t + 1$	g^{44}	$3t^2 + 3t + 4$	g^{76}	$4t^2 + 2t + 3$	g^{108}	$3t^2 + t + 4$
g^{13}	$t^2 + t + 3$	g^{45}	$3t^2 + 4t + 1$	g^{77}	$t^2 + 2t + 3$	g^{109}	$2t^2 + t + 3$
g^{14}	$t^2 + 3t + 2$	g^{46}	$2t^2 + 4t + 1$	g^{78}	$4t^2 + 2t + 1$	g^{110}	$3t + 1$
g^{15}	$4t^2 + 3t + 2$	g^{47}	$3t^2 + 4t + 2$	g^{79}	$t + 2$	g^{111}	$2t^2$
g^{16}	$t^2 + 3t + 4$	g^{48}	$2t + 4$	g^{80}	$4t^2$	g^{112}	$3t^2 + 3$
g^{17}	$4t + 3$	g^{49}	$3t^2$	g^{81}	$t^2 + 1$	g^{113}	$t^2 + 4t + 1$
g^{18}	t^2	g^{50}	$2t^2 + 2$	g^{82}	$2t^2 + 3t + 2$	g^{114}	$4t^2 + 4t + 3$
g^{19}	$4t^2 + 4$	g^{51}	$4t^2 + t + 4$	g^{83}	$3t^2 + 3t + 1$	g^{115}	$2t^2 + 1$
g^{20}	$3t^2 + 2t + 3$	g^{52}	$t^2 + t + 2$	g^{84}	$4t^2 + 2$	g^{116}	$t^2 + 3t + 1$
g^{21}	$2t^2 + 2t + 4$	g^{53}	$3t^2 + 4$	g^{85}	$2t^2 + t + 2$	g^{117}	$t^2 + 4$
g^{22}	$t^2 + 3$	g^{54}	$4t^2 + 2t + 4$	g^{86}	$2t^2 + 3$	g^{118}	$t^2 + 2t + 1$
g^{23}	$3t^2 + 4t + 3$	g^{55}	$4t^2 + 1$	g^{87}	$2t^2 + 4t + 2$	g^{119}	$3t^2 + t$
g^{24}	$3t^2 + 2$	g^{56}	$4t^2 + 3t + 4$	g^{88}	$t^2 + 2t$	g^{120}	$4t + 1$
g^{25}	$3t^2 + t + 3$	g^{57}	$2t^2 + 4t$	g^{89}	$3t + 2$	g^{121}	$4t + 4$
g^{26}	$4t^2 + 3t$	g^{58}	$t + 4$	g^{90}	$3t + 3$	g^{122}	$4t^2 + 3t + 3$
g^{27}	$2t + 3$	g^{59}	$t + 1$	g^{91}	$3t^2 + t + 1$	g^{123}	$4t^2 + t + 2$
g^{28}	$2t + 2$	g^{60}	$t^2 + 2t + 2$	g^{92}	$3t^2 + 2t + 4$	g^{124}	1
g^{29}	$2t^2 + 4t + 4$	g^{61}	$t^2 + 4t + 3$	g^{93}	2		
g^{30}	$2t^2 + 3t + 1$	g^{62}	4	g^{94}	$t^2 + t + 1$		
g^{31}	3	g^{63}	$2t^2 + 2t + 2$	g^{95}	$2t + 1$		
g^{32}	$4t^2 + 4t + 4$	g^{64}	$4t + 2$	g^{96}	$4t^2 + t + 1$		

Table 2: The elements of $GF(5^3)$