Markus Jakobsson
Moti Yung
Jianying Zhou (Eds.)

LNCS 3089

Applied Cryptography and Network Security

Second International Conference, ACNS 2004
Yellow Mountain, China, June 2004
Proceedings

Springer

Markus Jakobsson Moti Yung
Jianying Zhou (Eds.)

Applied Cryptography and Network Security

Second International Conference, ACNS 2004
Yellow Mountain, China, June 8-11, 2004
Proceedings

Springer

Volume Editors

Markus Jakobsson
RSA Laboratories
1203 Garden Street, Hoboken, NJ 07030, USA
E-mail: mjakobsson@rsasecurity.com

Moti Yung
Columbia University, Computer Science Department
New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Jianying Zhou
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
E-mail: jyzhou@i2r.a-star.edu.sg

Library of Congress Control Number: 2004106759

CR Subject Classification (1998): E.3, C.2, D.4.6, H.3-4, K.4.4, K.6.5

ISSN 0302-9743
ISBN 3-540-22217-0 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11014096 06/3142 5 4 3 2 1 0

Preface

The second International Conference on Applied Cryptography and Network Security (ACNS 2004) was sponsored and organized by ICISA (the International Communications and Information Security Association). It was held in Yellow Mountain, China, June 8–11, 2004. The conference proceedings, representing papers from the academic track, are published in this volume of the Lecture Notes in Computer Science (LNCS) of Springer-Verlag.

The area of research that ACNS covers has been gaining importance in recent years due to the development of the Internet, which, in turn, implies global exposure of computing resources. Many fields of research were covered by the program of this track, presented in this proceedings volume. We feel that the papers herein indeed reflect the state of the art in security and cryptography research, worldwide.

The program committee of the conference received a total of 297 submissions from all over the world, of which 36 submissions were selected for presentation during the academic track. In addition to this track, the conference also hosted a technical/industrial track of presentations that were carefully selected as well. All submissions were reviewed by experts in the relevant areas.

Starting from the first ACNS conference last year, ACNS has given best paper awards. Last year the best student paper award went to a paper that turned out to be the only paper written by a single student for ACNS 2003. It was Kwong H. Yung who got the award for his paper entitled "Using Feedback to Improve Masquerade Detection." Continuing the "best paper tradition" this year, the committee decided to select two student papers among the many high-quality papers that were accepted for this conference, and to give them best student paper awards. These papers are: "Security Measurements of Steganographic Systems" by Weiming Zhang and Shiqu Li, and "Evaluating Security of Voting Schemes in the Universal Composability Framework" by Jens Groth. Both papers appear in this proceedings volume, and we would like to congratulate the recipients for their achievements.

Many people and organizations helped in making the conference a reality. We would like to take this opportunity to thank the program committee members and the external experts for their invaluable help in producing the conference's program. We also wish to thank Thomas Herlea of KU Leuven for his extraordinary efforts in helping us to manage the submissions and for taking care of all the technical aspects of the review process. Thomas, single-handedly, served as the technical support committee of this conference! We extend our thanks also to the general chair Jianying Zhou (who also served as publication chair and helped in many other ways), the chairs of the technical/industrial track (Yongfei Han and Peter Landrock), the local organizers, who worked hard to assure that the conference took place, and the publicity chairs. We also thank the various

sponsoring companies and government bodies. Finally, we would like to thank all the authors who submitted papers to the conference.

April 2004 Markus Jakobsson and Moti Yung

ACNS 2004

Second International Conference on Applied Cryptography and Network Security

Yellow Mountain, China
June 8–11, 2004

Sponsored and organized by the

International Communications and Information Security Association (ICISA)

In co-operation with

MiAn Pte Ltd (ONETS), China
RSA Security Inc., USA
Ministry of Science and Technology, China
Yellow Mountain City Government, China

General Chair

Jianying Zhou ..... Institute for Infocomm Research, Singapore

Program Chairs

Markus Jakobsson ..... RSA Labs, USA
Moti Yung ..... Columbia University, USA

Program Committee

Masayuki Abe ..... NTT, Japan
N. Asokan ..... Nokia, Finland
Feng Bao ..... I2R, Singapore
Kijoon Chae ..... Ewha Women's Univ., Korea
Ed Dawson ..... QUT, Australia
Xiaotie Deng ..... City Univ. of HK, China
Philippe Golle ..... PARC, USA
Dieter Gollmann ..... TU Hamburg, Germany
Goichiro Hanaoka ..... Univ. of Tokyo, Japan
Els van Herreweghen ..... IBM, Zurich
Chi-Sung Laih ..... NCKU, Taiwan
Kwok-Yan Lam ..... Tsinghua Univ., China
Heejo Lee ..... Korea Univ., Korea

Pil Joong Lee ..... Postech, Korea
Helger Lipmaa ..... Helsinki Univ. of Tech., Finland
Javier Lopez ..... Univ. of Malaga, Spain
Charanjit Jutla ..... IBM T.J. Watson, USA
Hiroaki Kikuchi ..... Univ. of Tokai, Japan
Kwangjo Kim ..... Info. & Communication Univ., Korea
Wenbo Mao ..... HP Labs, UK
David Naccache ..... Gemplus, France
Chanathip Namprempre ..... Thammasat U., Thailand
Phong Nguyen ..... ENS, France
Adrian Perrig ..... Carnegie Mellon Univ., USA
Josef Pieprzyk ..... Macquarie University, Australia
Radha Poovendran ..... Univ. of Washington, USA
Tomas Sander ..... HP Labs, USA
Dawn Song ..... Carnegie Mellon Univ., USA
Julien Stern ..... Cryptolog International, France
Sal Stolfo ..... Columbia Univ., USA
Michael Szydlo ..... RSA Labs, USA
Wen-Guey Tzeng ..... NCTU, Taiwan
Shouhuai Xu ..... Univ. of Texas at San Antonio, USA
Bennet Yee ..... Google, USA
Yuliang Zheng ..... UNC Charlotte, USA

Chairs of Technical/Industrial Track

Yongfei Han ..... ONETS, China
Peter Landrock ..... Cryptomathic, Denmark

Publicity Chairs

Michael Szydlo ..... RSA Labs, USA
Guilin Wang ..... I2R, Singapore

Technical and Administrative Support

Thomas Herlea ..... KU Leuven, Belgium
Li Xue ..... ONETS, China

External Reviewers

Michel Abdalla, Nuttapong Attrapadung, Dan Bailey, Dirk Balfanz, Endre-Felix Bangerter, Alexandra Boldyreva, Colin Boyd, Eric Brier, Julien Brouchier, Sonja Buchegger, Christian Cachin, Jan Camenisch, Cedric Cardonnel, Haowen Chan, Xiaofeng Chen, Benoît Chevallier-Mames, Hung Chim, Jung-Hui Chiu, Jae-Gwi Choi, Chen-Kang Chu, Siu-Leung Chung, Andrew Clark, Scott Contini, Jean-Sébastien Coron, Yang Cui, Matthew Dailey,

Jean-François Dhem, Xuhua Ding, Glenn Durfee, Pasi Eronen, Chun-I Fan, Serge Fehr, Atsushi Fujioka, Eiichiro Fujisaki, Debin Gao, Philip Ginzboorg, Juanma Gonzalez-Nieto, Louis Goubin, Zhi Guo, Shin Seong Han, Yumiko Hanaoka, Helena Handschuh, Matt Henricksen, Sha Huang, Yong Ho Hwang, Tetsuya Izu, Moon Su Jang, Ari Juels, Burt Kaliski, Bong Hwan Kim, Byung Joon Kim, Dong Jin Kim, Ha Won Kim, Kihyun Kim, Tae-Hyung Kim, Yuna Kim, Lea Kissner, Tetsutaro Kobayashi, Byoungcheon Lee, Dong Hoon Lee, Hui-Lung Lee, Chin-Laung Lei, Jung-Shian Li, Mingyan Li, Minming Li, Tieyan Li, Becky Jie Liu, Krystian Matusiewicz, Bill Millan, Ilya Mironov, David M'Raïhi, Yasusige Nakayama, Gregory Neven, James Newsome, Valtteri Niemi, Takashi Nishi, Kaisa Nyberg, Luke O'Connor, Kazuto Ogawa, Miyako Ohkubo, Jose A. Onieva, Pascal Paillier, Dong Jin Park, Heejae Park, Jae Hwan Park, Joonhah Park, Leonid Peshkin, Birgit Pfitzmann, James Riordan, Rodrigo Roman, Ludovic Rousseau, Markku-Juhani Saarinen, Radha Sampigethaya, Paolo Scotton, Elaine Shi, Sang Uk Shin, Diana Smetters, Miguel Soriano, Jessica Staddon, Ron Steinfeld, Reto Strobl, Hong-Wei Sun, Koutarou Suzuki, Vanessa Teague, Lawrence Teo, Ali Tosun, Johan Wallen, Guilin Wang, Huaxiong Wang, Yuji Watanabe, Yoo Jae Won, Yongdong Wu, Yeon Hyeong Yang, Tommy Guoming Yang, Sung Ho Yoo, Young Tae Youn, Dae Hyun Yum, Rui Zhang, Xinwen Zhang, Hong Zhao, Xi-Bin Zhao, Yunlei Zhao, Huafei Zhu

Table of Contents

Security and Storage

CamouflageFS: Increasing the Effective Key Length in Cryptographic Filesystems on the Cheap 1
Michael E. Locasto, Angelos D. Keromytis

Private Keyword-Based Push and Pull with Applications to Anonymous Communication 16
Lea Kissner, Alina Oprea, Michael K. Reiter, Dawn Song, Ke Yang

Secure Conjunctive Keyword Search over Encrypted Data 31
Philippe Golle, Jessica Staddon, Brent Waters

Provably Secure Constructions

Evaluating Security of Voting Schemes in the Universal Composability Framework 46
Jens Groth

Verifiable Shuffles: A Formal Model and a Paillier-Based Efficient Construction with Provable Security 61
Lan Nguyen, Rei Safavi-Naini, Kaoru Kurosawa

On the Security of Cryptosystems with All-or-Nothing Transform 76
Rui Zhang, Goichiro Hanaoka, Hideki Imai

Internet Security

Centralized Management of Virtual Security Zones in IP Networks 91
Antti Peltonen, Teemupekka Virtanen, Esa Turtiainen

S-RIP: A Secure Distance Vector Routing Protocol 103
Tao Wan, Evangelos Kranakis, Paul C. van Oorschot

A Pay-per-Use DoS Protection Mechanism for the Web 120
Angelos Stavrou, John Ioannidis, Angelos D. Keromytis, Vishal Misra, Dan Rubenstein

Digital Signature

Limited Verifier Signature from Bilinear Pairings 135
Xiaofeng Chen, Fangguo Zhang, Kwangjo Kim

Deniable Ring Authentication Revisited 149
Willy Susilo, Yi Mu

A Fully-Functional Group Signature Scheme
over Only Known-Order Group 164
Atsuko Miyaji, Kozue Umeda

Security Modelling

Some Observations on Zap and Its Applications 180
Yunlei Zhao, C.H. Lee, Yiming Zhao, Hong Zhu

Security Measurements of Steganographic Systems 194
Weiming Zhang, Shiqu Li

X^{2}Rep: Enhanced Trust Semantics for the XRep Protocol 205
Nathan Curtis, Rei Safavi-Naini, Willy Susilo

Authenticated Key Exchange

One-Round Protocols for Two-Party Authenticated Key Exchange 220
Ik Rae Jeong, Jonathan Katz, Dong Hoon Lee

Password Authenticated Key Exchange Using Quadratic Residues 233
Muxiang Zhang

Key Agreement Using Statically Keyed Authenticators 248
Colin Boyd, Wenbo Mao, Kenneth G. Paterson

Security of Deployed Systems

Low-Latency Cryptographic Protection for SCADA Communications 263
Andrew K. Wright, John A. Kinast, Joe McCarty

A Best Practice for Root CA Key Update in PKI 278
InKyoung Jeun, Jongwook Park, TaeKyu Choi, SangWan Park, BaeHyo Park, ByungKwon Lee, YongSup Shin

SQLrand: Preventing SQL Injection Attacks 292
Stephen W. Boyd, Angelos D. Keromytis

Cryptosystems: Design and Analysis

Cryptanalysis of a Knapsack Based Two-Lock Cryptosystem 303
Bin Zhang, Hongjun Wu, Dengguo Feng, Feng Bao

Success Probability in χ^2-Attacks 310
Takashi Matsunaka, Atsuko Miyaji, Yuuki Takano

More Generalized Clock-Controlled Alternating Step Generator 326
Ali A. Kanso

Cryptographic Protocols

FDLKH: Fully Decentralized Key Management Scheme on Logical Key Hierarchy 339
Daisuke Inoue, Masahiro Kuroda

Unconditionally Non-interactive Verifiable Secret Sharing Secure against Faulty Majorities in the Commodity Based Model 355
Anderson C.A. Nascimento, Joern Mueller-Quade, Akira Otsuka, Goichiro Hanaoka, Hideki Imai

Cryptanalysis of Two Anonymous Buyer-Seller Watermarking Protocols and an Improvement for True Anonymity 369
Bok-Min Goi, Raphael C.-W. Phan, Yanjiang Yang, Feng Bao, Robert H. Deng, M.U. Siddiqi

Side Channels and Protocol Analysis

Security Analysis of CRT-Based Cryptosystems 383
Katsuyuki Okeya, Tsuyoshi Takagi

Cryptanalysis of the Countermeasures Using Randomized Binary Signed Digits 398
Dong-Guk Han, Katsuyuki Okeya, Tae Hyun Kim, Yoon Sung Hwang, Young-Ho Park, Souhwan Jung

Weaknesses of a Password-Authenticated Key Exchange Protocol between Clients with Different Passwords 414
Shuhong Wang, Jie Wang, Maozhi Xu

Intrusion Detection and DoS

Advanced Packet Marking Mechanism with Pushback for IP Traceback 426
Hyung-Woo Lee

A Parallel Intrusion Detection System for High-Speed Networks 439
Haiguang Lai, Shengwen Cai, Hao Huang, Junyuan Xie, Hui Li

A Novel Framework for Alert Correlation and Understanding 452
Dong Yu, Deborah Frincke

Cryptographic Algorithms

An Improved Algorithm for $uP + vQ$ Using JSF_3^1 467
BaiJie Kuang, YueFei Zhu, YaJuan Zhang

New Table Look-Up Methods for Faster Frobenius Map Based Scalar Multiplication Over $GF(p^n)$ 479
Palash Sarkar, Pradeep Kumar Mishra, Rana Barua

Batch Verification for Equality of Discrete Logarithms and Threshold Decryptions 494
Riza Aditya, Kun Peng, Colin Boyd, Ed Dawson, Byoungcheon Lee

Author Index 509

CamouflageFS: Increasing the Effective Key Length in Cryptographic Filesystems on the Cheap

Michael E. Locasto and Angelos D. Keromytis

Department of Computer Science
Columbia University in the City of New York
{locasto,angelos}@cs.columbia.edu

Abstract. One of the few quantitative metrics used to evaluate the security of a cryptographic file system is the key length of the encryption algorithm; larger key lengths correspond to higher resistance to brute force and other types of attacks. Since accepted cryptographic design principles dictate that larger key lengths also impose higher processing costs, increasing the security of a cryptographic file system also increases the overhead of the underlying cipher.
We present a general approach to effectively extend the key length without imposing the concomitant processing overhead. Our scheme is to spread the ciphertext inside an artificially large file that is seemingly filled with random bits according to a key-driven spreading sequence. Our prototype implementation, *CamouflageFS,* offers improved performance relative to a cipher with a larger key-schedule, while providing the same security properties. We discuss our implementation (based on the Linux Ext2 file system) and present some preliminary performance results. While CamouflageFS is implemented as a stand-alone file system, its primary mechanisms can easily be integrated into existing cryptographic file systems.

"Why couldn't I fill my hard drive with random bytes, so that individual files would not be discernible? Their very existence would be hidden in the noise, like a striped tiger in tall grass." –Cryptonomicon, by Neal Stephenson [17]

1 Introduction

Cryptographic file systems provide data confidentiality by employing encryption to protect files against unauthorized access. Since encryption is an expensive operation, there is a trade-off between performance and security that a system designer must take into consideration. One factor that affects this balance is the key length of the underlying cipher: larger key lengths imply higher resistance against specific types of attacks, while at the same time requiring more rounds of processing to spread the influence of the key across all plaintext bit ("avalanche effect"). This is by no means a clear-cut comparison, however: different ciphers can exhibit radically different performance characteristics (*e.g.,* AES with 128 bit keys is faster than DES with 56 bit keys), and the security of a cipher is not simply encapsulated by its key length. However, given a well designed variable-key length cryptographic cipher, such as AES, the system designer or administrator is faced with the balance of performance *vs.* key length.

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 1–15, 2004.

We are interested in reducing the performance penalty associated with using larger key sizes without decreasing the level of security. This goal is accomplished with a technique that is steganographic in nature; we camouflage the parts of the file that contain the encrypted data. Specifically, we use a spread-spectrum code to distribute the pointers in the file index block. We alter the operating system to intercept file requests made without an appropriate key and return data that is consistently random (*i.e.*, reading the same block will return the same "garbage"), without requiring that such data be stored on disk. This random data is indistinguishable from encrypted data. In this way, each file appears to be an opaque block of bits on the order of a terabyte. There is no need to actually fill the disk with random data, as done in [13], because the OS is responsible for generating this fake data on the fly. An attacker must mount a brute force attack not only against the underlying cipher, but also against the spreading sequence. In our prototype, this can increase an attacker's work factor by 2^{28} without noticeable performance loss for legitimate users.

1.1 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we discuss our approach to the problem, examine the threat model, and provide a security analysis. In Section 3 we discuss in detail the implementation of CamouflageFS as a variant of the Linux Ext2fs, and Section 4 presents some preliminary performance measurements of the system. We give an overview of the related work on cryptographic and steganographic file systems in Section 5. We discuss our plans for future work in Section 6, and conclude the paper in Section 7.

2 Our Approach

Our primary insight is that a user may decrease the performance penalty they pay for employing a cryptographic file system by using only part of the key for cryptographic operations. The rest of the key may be used to unpredictably spread the data into the file's address space. Note that we are not necessarily fragmenting the placement of the data on disk, but rather mixing the placement of the data within the file.

2.1 Key Composition: Maintaining Confidentiality

While our goal is to mitigate the performance penalty paid for using a cryptographic file system, it is not advisable to trade confidentiality for performance. Instead, we argue that keys can be made effectively longer *without* incurring the usual performance penalty. One obvious method of reducing the performance penalty for encrypting files is to utilize a cipher with a shorter key length; however, there is a corresponding loss of confidentiality with a shorter key length. We address the tradeoff between key length and performance by extending the key with "spreading bits," and exploiting the properties of an indexed allocation file system.

A file system employing indexed allocation can efficiently address disk blocks for files approaching terabyte size. In practice, most files are much smaller than this and do

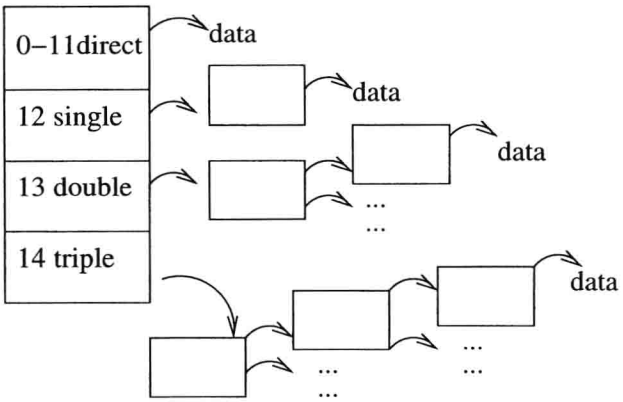

Fig. 1. Outline of a multi-level index scheme with triple-indirect addressing. The first 12 index entries point directly to 12 data blocks. The next three index entries are single, double, and triple indirect. Each indirect block contains 1024 entries: the first level can point to 1024 data blocks, the second level can point to 1024^2, and the third level points to 1024^3 data blocks.

not use their full "address space." The Linux Ext2fs on 32-bit architectures commonly provides an address range of a few gigabytes to just short of two terabytes, depending on the block size, although accessing files larger than two gigabytes requires setting a flag when opening the file [4].

We use the extra bits of the cryptographic key to spread the file data throughout its address space and use the primary key material to encrypt that data. By combining this spreading function with random data for unallocated blocks, we prevent an attacker from knowing which blocks to perform a brute force search on. To maintain this illusion of a larger file without actually allocating it on disk, we return consistently random data on *read()* operations that are not accompanied by the proper cryptographic key.

2.2 Indexed Allocation

In a multi-level indexed allocation scheme, the operating system maintains an index of entries per file that can quickly address any given block of that file. In the Ext2 file system, this index contains fifteen entries (see Figure 1). The first twelve entries point directly to the first twelve blocks of the file. Assuming a block size of 4096 bytes, the first twelve entries of this index map to the first 48Kb of a file. The next three entries are all indirect pointers to sub-indices, with one layer of indirection, two layers of indirection, and three layers of indirection, respectively [4].

Figure 2 shows a somewhat simplified example of a single-level direct-mapped index. The file index points directly to blocks with plaintext data. Holes in the file may exist; reading data from such holes returns zeroed-out blocks, while writing in the holes causes a physical disk block to be allocated. Cryptographic file systems encrypt the stored data, which leaves the index structure identical but protects the contents of the data blocks, as shown in Figure 3.

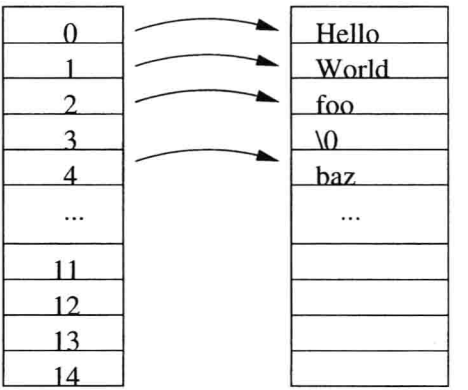

Fig. 2. File index for a normal data file. Pointers to plaintext data blocks are stored sequentially at the beginning of the index. Files may already contain *file holes* – this index has a hole at the third block position.

Usually, most files are small and do not need to expand beyond the first twelve direct mapped entries. This design allows the data in a small file to be retrieved in two disk accesses. However, retrieving data pointed to by entries of the sub-indices is not prohibitively expensive, especially in the presence of disk caches [4].

Therefore, instead of clustering the pointers to file data in the beginning entries of the index, we can distribute them throughout the index. In order for the operating system to reliably access the data in the file, we need some sequence of numbers to provide the *spreading schedule*, or which index entries point to the different blocks of the file. Figure 4 shows encrypted data that has been spread throughout the file's address space.

2.3 Spreading Schedule

The purpose of the *spreading schedule* is to randomly distribute the real file data throughout a large address space so that an attacker would have to first guess the spreading schedule before he attempts a brute force search on the rest of the key.

Normally, the number of the index entry is calculated by taking the floor of the current file position "pos" divided by the block size.

$$index = pos/blocksize$$

This index number is then used to derive the *logical block number* (the block on disk) where the data at "pos" resides.

$$lbn = get\_from\_index(index)$$

This procedure is altered to employ the spreading schedule. The initial calculation of the index is performed, but before the logical block number is derived, a pseudo-random permutation (PRP) function takes the calculated index and the bits of the spreading seed

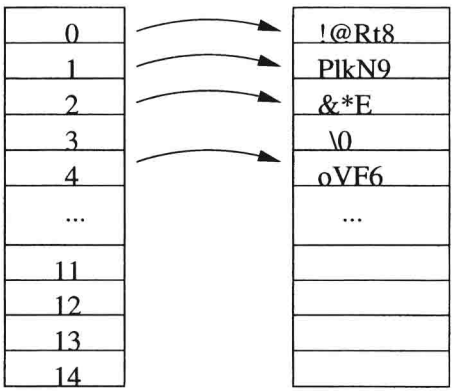

Fig. 3. Index for an encrypted file. The indexing has not changed, merely the contents of the data blocks. Again, the file hole at block three is present.

to return a new index value, without producing collisions. The logical block number is then derived from this new index.

$$index = pos/blocksize$$

$$index = map(index, spread\_seed)$$

$$lbn = get\_from\_index(index)$$

Note that the actual disk block is irrelevant; we are only interested in calculating a new entry in the file index, rather than using the strictly sequential ordering. Given the secret spreading seed bits of the key, this procedure will return consistent results. Therefore, using the same key will produce a consistent spreading schedule, and a legitimate user can easily retrieve and decrypt their data.

2.4 Consistent Garbage

The spreading schedule is useless without some mechanism to make the real encrypted data appear indistinguishable from unallocated data blocks. To accomplish this blending, camouflage data is generated by the operating system whenever a request is made on an index entry that points to unallocated disk space (essentially a file hole). Each CamouflageFS file will contain a number of file holes. Without the key, a request on any index entry will return random data. There is no way to determine if this data is encrypted without knowing the spreading schedule, because data encrypted by a strong cipher should appear to be random in its ciphertext form. We employ a linear congruential generator [11] (LCG) to provide pseudo-random data based on a secret random quantity known only to the operating system. This final touch camouflages the actual encrypted data, and the file index is logically similar to Figure 5. Note that camouflage data is only needed (and created on the fly) when the system is under attack; it has no impact on performance or disk capacity under regular system operation.