Dependable and Secure Data Storage in Wireless Ad Hoc Networks: An Assessment of DS^2

S. Chessa[1,2], R. Di Pietro[3], and P. Maestrini[1,2]

[1]Dipartimento di Informatica, Università di Pisa,
via Buonarroti 2, 56127 Pisa, Italy
[2]Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo",
Area della Ricerca CNR, via Moruzzi 1, 56124 Pisa, Italy
[3]Dipartimento di Informatica, Università di Roma "La Sapienza",
via Salaria, 113 – 00198 Roma, Italy

Abstract. DS^2 is a dependable and secure data storage for mobile, wireless networks based on a peer-to-peer paradigm. DS^2 provides support to create and share files under a write-once model, and ensures data confidentiality and dependability by encoding files in a Redundant Residue Number System. The paper analyzes the code efficiency of DS^2 using a set of moduli allowing for efficient encoding and decoding procedures based on single precision arithmetic, and discusses the security issues. The comparison of DS^2 with the Information Dispersal Algorithm approach (IDA) shows that DS^2 features security features which are not provided by IDA, while the two approaches are comparable from the viewpoint of code efficiency and encoding/decoding complexity.

1 Introduction

Mobile ad hoc networks are composed by a set of mobile hosts (also called mobiles) communicating with each other via radio transceivers. In order to communicate with destinations which are located outside of their transmission ranges or hidden by obstacles, communicating mobiles rely on other mobiles which cooperate to forward messages to their destinations. To this purpose the network layer of the mobiles provides services of message delivery by running suitable routing algorithms [1], [2]. However, mobility and failures may give rise to network disconnections impairing service dependability.

Due to mobility of nodes, the network topology varies with time. At a given instant of time it is described by a graph where nodes are the mobiles, and a link connecting two nodes in the graph means that the corresponding mobiles can communicate directly.

The mobiles rely on on-board batteries for energy supply, hence energy efficiency of mobiles is an important issue [3]. The effect of battery depletion is similar to a crash fault, from which the mobile may or may not recover depending on the avail-

R. Battiti, M. Conti, and R. Lo Cigno (Eds.): WONS 2004, LNCS 2928, pp. 184–198, 2004.

ability of battery replacement/recharge. As mobiles may not be equipped with permanent storage, failures may result in data losses or corruption.

An important issue in mobile ad hoc networks is how to implement dependable and secure data storage. This is an essential requirement in applications where the mobiles cooperate by sharing information and need to create and access shared files. The system should prevent data losses or corruption due to network disconnections, mobile failures or malicious attacks from untrustworthy mobiles, and it should provide the file owners with mechanisms for secure distribution of files access privileges.

Several techniques to implement dependable and/or secure data storage have been proposed in the recent literature [4-10]. Some of these, which are based on a client-server paradigm [4], [5], hardly fit the ad hoc network model which is rather based on a peer to peer paradigm. Other approaches are conceived for systems connected with fast, wired networks, where mobility and disconnections of nodes is not supported [6], or which pay considerable communication overhead to implement a sophisticate model of intrusion tolerance based on user authentication [7].

Dependable storage systems based on a peer to peer paradigm which may adapt to the ad hoc network model have also been introduced [8], [9]. They exploit techniques of data fragmentation and dispersal [10] based on erasure codes [11] and use cryptography to achieve data confidentiality.

The technique of data fragmentation and dispersal was originally introduced in [10], where an information dispersal algorithm (IDA) had been proposed. It exploits erasure codes which are optimal with respect to code efficiency and allows for efficient encoding and decoding procedures.

A new technique to achieve dependable and secure data storage (DS^2) in wireless networks has been proposed in [12]. DS^2 exploits *Redundant Residue Number System* (*RRNS*) [13], [14] to encode data, which allows for a uniform coverage of both erasure and errors. *RRNS* encode data as (h+r)-tuples of residues using h+r keys, or moduli. Residues are distributed among the mobiles in the network. Recovering the original information requires the knowledge of at least h residues and of the corresponding moduli. Data can be reconstructed in the presence of up to $s \leq r$ residue losses (erasures), combined with up to $\left\lfloor \frac{r-s}{2} \right\rfloor$ corrupted residues. As compared to IDA, DS^2 features basic data confidentiality which is inherently provided by the *RRNS* encoding. Data confidentiality is ensured since mobiles having access to the residues are able to decode them only if they also know the correspondence of the residues with the set of moduli.

In this paper we evaluate the code efficiency and the coding/decoding complexity of DS^2, we analyze the security issues related to DS^2 and we compare DS^2 and IDA.

The rest of the paper is organized as follows. The Redundant Residue Number Systems and DS^2 are reviewed in sections 2 and 3, respectively. Section 4 and 5 analyze the code efficiency and the encoding/decoding complexity, respectively. Section 6 discusses the security issues and Section 7 compares DS^2 with IDA. Section 8 draws the conclusions.

2 Redundant Residue Number Systems

Given $h+r$ pairwise prime, positive integers $m_1,\ldots,m_{h+r}$ called *moduli*, let $M = \prod_{p=r+1}^{h+r} m_p$, $M_R = \prod_{p=1}^{r} m_p$, and, without loss of generality, $m_p > m_{p-1}$ for each $p \in [2,h]$. Given any non-negative integer X, let $x_p = X \bmod m_p$ be the residue of X modulo m_p. In the rest of the paper also notation $(a)_b$ will also be used to denote $a \bmod b$.

The number system representing integers in $[0,M)$ with the $(h+r)$-tuples of their residues modulo $m_1,\ldots,m_{h+r}$ is called the *Redundant Residue Number System* (*RRNS*) of moduli $m_1,\ldots,m_{h+r}$, range M and redundancy M_R [13], [14]. For every $h+r$-tuple $(x_1,\ldots,x_{h+r})$, the corresponding integer X can be reconstructed by means of the Chinese Remainder Theorem:

$$X = \left(\sum_{p=1,h+r} \left(\frac{M \cdot M_R}{m_p} \left(x_p \beta_p \right)_{m_p} \right) \right)_{M \cdot M_R} \tag{1}$$

where, for each $p \in [1,h]$, $\beta_p = \left\langle \frac{M \cdot M_R}{m_p} \right\rangle_{m_p}$ is the multiplicative inverse of $M \cdot M_R / m_p$ modulo m_p, that is, $\left(\frac{M \cdot M_R}{m_p} \beta_p \right)_{m_p} = 1$ [15], and β_p is in the range $[0,m_p)$.

Although the given *RRNS* could provide unique representations to all integers in the range $[0,M \cdot M_R)$ [15], the legitimate range of representation is limited to $[0,M)$, and the corresponding $h+r$-tuples, are called *legitimate*. Integers in $[M,M \cdot M_R)$ and the corresponding $(h+r)$-tuples are called *illegitimate*.

Given an *RRNS* of range M and redundancy M_R, with moduli $m_1,\ldots,m_{h+r}$, let $(x_1,\ldots, x_{h+r})$ be the legitimate representation of some X in $[0,M)$. An *erasure* of multiplicity e is an event making unavailable e arbitrary digits in the representation, and an *error* of multiplicity d is an event transforming d arbitrary, unknown digits. If $e+2d \leq r$ then the *RRNS* can correct the errors to reconstruct X [12].

Efficient error correcting algorithms are reported in [16-18], while an overview on *RRNS* is available in [19].

3 The Dependable and Secure Data Storage for Ad Hoc Networks (DS^2)

In the Dependable and Secure Data storage for mobile ad hoc networks (DS^2) [12] the mobiles cooperate by creating and sharing files. The system provides procedures to create, share and access the files. Once created a file can be written or removed only by its owner (the file creator).

Hereafter we assume that each mobile is assigned with a unique identifier ranging from 0 to n–1, and we will sometimes use the concise notation u_i to denote the mobile i.

The file creation procedure exploits an appropriate *RRNS* to encode a file. To this purpose u_i selects a set of h+r moduli (pairwise prime positive integers) $m_1,\ldots,m_{h+r}$ with $m_p > m_{p-1}$ for each $p \in [2,h+r]$. The moduli are chosen among a set of available moduli computed offline. Since the maximum number which can be represented is limited by the range $M=m_{r+1}\cdot\ldots\cdot m_{h+r}$ of the *RRNS*, files of sizes exceedingly the range are preliminary partitioned into records $b_1,\ldots,b_s$ of size b bits each, with $2^b \leq M$, and each record is encoded separately.

Record b_t is encoded in the *RRNS* of moduli $m_1,\ldots,m_{h+r}$ by the (h+r)-tuple of residues $(x_{t,1},\ldots,x_{t,h+r})$. Each residue is sent to a different mobile currently reachable by the file creator, which in turn stores the residue in its storage. The assignment of mobiles to residues is arbitrary with the only constraint that different residues of the same record should be stored in different mobiles. Note that the mobiles storing the residue are not provided with any information about the modulo used to encode that residue.

The file owner maintains a file descriptor containing the set of the moduli used for encoding and *s record descriptors*. Each record descriptor contains the set of the list mobiles storing the residue digits of the record with the correct association between mobiles and residues. The file descriptor is kept secret by the owner.

Due to the encoding properties, the file records can be read separately. Record reading requires knowledge of correspondence between the mobiles storing the residue and the moduli used to encode the residues. Assuming that a mobile i owns a copy of the file descriptor, it can issue read requests to all the mobiles storing a given record b_t. During the read procedure some of the requested residues could be lost during the communication or even could be corrupted before reaching i. Once u_i receives a sufficient number of residues, it executes the decoding procedure based on the Chinese Reminder Theorem. If no residues are corrupted the decoding procedure returns the value of b_t, otherwise u_i will attempt to recover from the corrupted residues. In general, the original content of record b_t can be recovered only if the residues are decoded with the correct moduli and the multiplicity of the erasures e and of the errors d is such that $2d+e \leq r$.

File sharing is enabled by the file owner by distributing encrypted replicas of the file descriptor to trusted mobiles. In DS^2 it is assumed that mobiles sharing a file are not allowed to distribute the file descriptor to other mobiles, nor to write or remove the file. Since it is assumed that the mobiles sharing the file are trusted by the file owner, DS^2 does not employ any mechanism to inhibit distribution of descriptors. The policy of denying write and remove privileges is enforced by the mobiles hosting the residue digits. Failure to enforce this policy is equivalent to malicious digit corruption by the hosts.

The reader is referred to [12] for an extensive presentation of the DS^2.

Table 1. Set of moduli used for the encoding

m_1	65536	m_{10}	65503	m_{19}	65449	m_{28}	65393	m_{37}	65339
m_2	65533	m_{11}	65501	m_{20}	65447	m_{29}	65383	m_{38}	65327
m_3	65531	m_{12}	65497	m_{21}	65437	m_{30}	65381	m_{39}	65323
m_4	65529	m_{13}	65491	m_{22}	65431	m_{31}	65371	m_{40}	65321
m_5	65527	m_{14}	65489	m_{23}	65423	m_{32}	65369	m_{41}	65311
m_6	65525	m_{15}	65479	m_{24}	65419	m_{33}	65363	m_{42}	65309
m_7	65521	m_{16}	65477	m_{25}	65413	m_{34}	65357	m_{43}	65293
m_8	65519	m_{17}	65473	m_{26}	65411	m_{35}	65353	m_{44}	65287
m_9	65509	m_{18}	65459	m_{27}	65407	m_{36}	65347	m_{45}	65281

4 Code Efficiency of DS^2

We firstly evaluate the code efficiency without redundancy (that is, r=0). Let f be a file created by mobile i composed by s records $b_1,\ldots,b_s$, where each record consists of b bits.

To improve the performance of the encoding/decoding procedures, we select the set of moduli such that most of the operations can be performed using single precision arithmetic. To this purpose we constructed a library of 45 moduli which is shown in Table 1. The largest modulo m_1 is 2^{16}, and all the other moduli have been chosen as close as possible to 2^{16}, with the constraint that the moduli must be pairwise prime.

Consider the Residue Number System with no redundancy (thus r=0) of the first h moduli of Table 1, and let $M= m_1\cdot\ldots\cdot m_h$ be its range and $b=\lfloor \log_2 M \rfloor$. Assume that record b_t ($t\in[1,s]$) is in the range $[0,2^b)$ (it can be represented with b bits), and b_t is encoded into the set of residues $x_{t,1},\ldots,x_{t,h}$ where residue $x_{t,p}$ ($p\in[1,h]$) is encoded with 16 bits. Hence the entire record is encoded in e=16h bits.

The code efficiency is defined as the ratio $\varphi=b/e$. The code efficiency of encoding with the first h moduli in the library of Table 1 is above 0.96 for $h\leq6$, and it is above 0.99 for $h>6$.

However, assuming range $[0,2^b)$ for records, would imply that the length of the records, prior to residue encoding, would not be aligned to bytes. If this is a requirement, the length of records should be set to $b'=b-\beta'$, where β is the smallest positive integer such that b' is a multiple of the byte length. For any choice of h in the library of Table 1, with $1<h\leq45$, $b'=16h-8$, and the record length b' scales linearly with h.

Under the latter assumption, Figure 1a depicts the code efficiency in terms of the ratio $\varphi=b'/e$ for different values of h. It is seen that, as h increases, the difference between b' and e remains constant while b' and e increase, and hence the code efficiency also increases.

We now evaluate the code efficiency using r>0 redundant moduli. Let us consider the *RRNS* of the first h+r moduli of Table 1. Since the redundant moduli should be larger than the non-redundant ones, the range is given by $M=m_r\cdot\ldots\cdot m_{h+r}$, and and $b=\lfloor \log_2 M \rfloor$. Record b_t ($t\in[1,s]$) is encoded into the set of residues $x_{t,1},\ldots,x_{t,h+r}$, where

$x_{t,p}$ ($p\in[1,h+r]$) has length 16 bits, and the length of the encoded record is $e=16(h+r)$. Assuming that the length of the record, prior to residue encoding is aligned to the byte, and defining $b'=16h-8$. as above, the code efficiency $\varphi=b'/e$ has been evaluated for $r\in[1,10]$ and $h\in[1,35]$. As shown in Figure 1b the code efficiency increases rapidly for $h<10$, after which it asymptotically approaches $h/(h+r)$.

5 Encoding/Decoding Complexity of DS^2

Consider first the complexity of operations $(p+q)_m$ and $(pq)_m$, with $p,q\in[0,2^l)$. For the sake of simplicity, we tolerate that the computation yields $[p+q]_m$ and $[pq]_m$ where $[x]_m$ denotes an integer in $[0,2^l)$ congruent to $(x)_m$. This substitution is tolerable, and sometimes useful, in the application under consideration. It is assumed that $m=2^l-\delta$, with $l=16$ and $\delta<2^8$. The latter assumption holds for all moduli in the library of Table 1.

Given integers $p,q\in[0,2^l)$, let $p+q=a_1 2^l+b_1$ where a_1 and b_1 are non negative integers. From $p+q\leq 2^l+2^l-1$ and $m=2^l-\delta$ it is immediate that $a_1\leq 1$, $b_1\leq 2^l-1$ and $[p+q]_m=[a_1 2^l+b_1]_m=[a_1 m+a_1\delta+b_1]_m=[a_1\delta+b_1]_m$. If $a_1=0$ then the single precision modulo is obtained. Else ($a_1=1$) let $\delta+b_1=a_2 2^l+b_2$ with a_2 and b_2 non negative integers and $a_2\leq 1$. Observe that $a_2=1$ implies $b_2=\delta+b_1-2^l\leq\delta+2^l-1-2^l=\delta-1$. Then $[\delta+b_1]_m=[a_2 2^l+b_2]_m=[a_2 m+a_2\delta+b_2]_m=[a_2\delta+b_2]_m$. From $b_2\leq\delta-1<m$ follows that $[\delta+b_1]_m=\delta+b_2$ if $a_2=1$ and $[\delta+b_1]_m=b_2$ if $a_2=0$. Then the complexity of $[\delta+b_1]_m$ requires a single precision addition, and computing $[p+q]_m$ requires two single precision additions in the worst case.

Similarly, given integers $p,q\in[0,2^l)$, let $pq=a_1 2^l+b_1$ where a_1 and b_1 are non negative integers. From $pq\leq 2^l(2^l-2)+1$, it is immediate that $a_1\leq 2^l-2$, $b_1\leq 2^l-1$, and $[pq]_m=[a_1 m+a_1\delta+b_1]_m=[a_1\delta+b_1]_m=[[a_1\delta]_m+b_1]_m$. Letting $a_1\delta=a_2 2^l+b_2$ with a_2 and b_2 non negative integers, from $a_1\delta\leq(2^l-2)\delta\leq 2^l(\delta-1)+2^l-\delta$, it follows $a_2\leq\delta-1$. In turn, $[a_1\delta]_m=[a_2 m+a_2\delta+b_2]=[a_2\delta+b_2]_m$. Since $a_2\delta\leq(\delta-1)\delta<2^l$ and $b_2<2^l$, computing $[a_1\delta]_m$ requires 2 single precision additions in the worst case.

In conclusion, computing $[pq]_m$ requires at most 2 single precision multiplications (to yield the pairs (a_1,b_1) and (a_2,b_2)) and 4 single precision additions (2 additions to yield $[a_1\delta]_m$ as $[a_2\delta+b_2]_m$ and 2 additions to yield $[pq]_m$ as $[[a_1\delta]_m+b_1]_m$).

Given an *RRNS* of the first $h+r$ moduli $m_1,\ldots,m_{h+r}$ of Table 1 and range $M=m_r\cdot\ldots\cdot m_{h+r}$, consider now the complexity of the procedure of residue encoding of an integer X in the range $[0,M)$. X is expanded as:

$$X=\sum_{i=0}^{h-1}\left(a_i 2^{il}\right) \tag{1}$$

The encoding procedure computes residues $x_1,\ldots,x_{h+r}$, where residue $x_i=\left(X\right)_{m_i}$ for each $i\in[1,h+r]$. For the ease of notation, let $m=m_i$, $\delta=\delta_i$ for some $i\in[1,h+r]$. From (1) and $m=2^l-\delta$ follows that $x=\left(\sum_{i=0}^{h-1}\left(a_i\left(m+\delta\right)^i\right)\right)_m=\left(\sum_{i=0}^{h-1}\left(a_i\delta^i\right)_m\right)_m$, and then

$$x = \left(\sum_{i=0}^{h-1} \left(a_i \left(\delta^i \right)_m \right)_m \right)_m \tag{2}$$

The values of $(\delta^i)_m$ can be computed offline and stored with the moduli in the library of Table 1: this requires storing 4h single precision integers for each modulo. It is immediate from (2) that the computation of each residue requires at most h multiplications mod m and h–1 additions mod m, that is, 2h single precision integer multiplications and 4h+2h–2=6h–2 single precision integer additions in the worst case.

In order to evaluate the complexity of decoding, assume without loss of generality that $h+t$ ($0 \le t \le r$) residues $x_1,\ldots,x_{h+t}$ from the encoding of a given record are received correctly, and let $M'=m_1 \cdot \ldots \cdot m_{h+t}$. Then

$$X = \left(\sum_{i=1,h+t} \left(\frac{M'}{m_i} \left(x_i \beta_i \right)_{m_i} \right) \right)_{M'} \tag{3}$$

where $\beta_i = \left\langle \frac{M'}{m_i} \right\rangle_{m_i}$ ($i \in [1,h+t]$) is a single precision integer since $\beta_i < m_i$. From (3) the decoding procedure involves the following operations:

1. $h+t$ multiple precision multiplications to yield $M'/m_i \left(x_i \beta_i \right)_{m_i}$;
2. $h+t-1$ multiple precision modular additions to yield the sum of the above products;
3. $h+t$ modular multiplications of single precision integers (that is, 2($h+t$) single precision multiplications and 4($h+t$) single precision integer additions) to yield $\left(x_i \beta_i \right)_{m_i}$ for each i;
4. computation of β_i for each i.

Regarding 4), the multiplicative inverse β_i can be efficiently computed if the multiplicative inverse of MM_R/m_i is known as a constant for every i. Such single precision integer can be computed offline for each modulo in the library of Table 1.

It is easily seen that β_i is given by:

$$\beta_i = \left(\prod_{p=h+t+1}^{h+r} \left(\left\langle \frac{MM_R}{m_i} \right\rangle_{m_i} m_p \right)_{m_i} \right)_{m_i} \tag{4}$$

Evaluation of Equation 4 requires $r-t$ modular single precision multiplications, that is, in the worst case 2($r-t$) single precision integer multiplications and 4t single precision, integer additions.

Considering that each multiple precision operation requires O($h+t$) operations, complexity of decoding $h+t$ residues has complexity O(($h+t$)^2).

6 Security Issues in DS^2

Confidentiality, authenticity and availability are among the classical security requirements. Confidentiality implies that only authorised users should be able to read a message; integrity implies that not authorized users are unable to modify a message, and the availability requirement consists in the protocol capacity to detect and resist to Denial of Service (DoS) attacks. In the following we analyse the compliance of DS^2 to such requirements.

Confidentiality. To recover a single record encoded into h+r residues by DS^2 (where h is the number of non-redundant moduli and r is the number of redundant moduli), an attacker must know the nodes on which the residues of the record are stored and the correspondence between each residue and the appropriate modulo. Further, the residues of a record do not provide information about the record content. More specifically a record can be successfully read only if the multiplicity of the erasures e and of the errors d is such that $2d+e\leq r$ and the correspondence between available residues and the moduli is known. Under these assumptions the record content can be recovered using using the Chinese Reminder Theorem. We are unaware of any efficient method to perform decoding which does not use the Chinese Reminder Theorem. However, the record can still be recovered by a brute force attack.

We evaluate the complexity of a brute force attack to decode the information of a record in the case in which h+t correct residues $\{x_1,\ldots,x_{h+t}\}$ are known. In principle an attacker may not know the values of h and r, however these indexes could be easily guessed by the attacker as the number of reasonable combinations is extremely limited, for this reason we assume that also h and r are known.

The attacker should consider all the possible h+t-tuples of residues in association with all the permutations of the h+r available moduli (which are public), and look for one combination leading to a legitimate number. Note that it is possible that several combinations lead to wrong legitimate numbers, but we disregard this possibility to the advantage of the attacker. It is then easy to see that the number of possible combinations is given by

$$\Theta = \binom{h+r}{h+t}(h+t)! = \frac{(h+r)!}{(r-t)!} \qquad (5)$$

It should be considered however that the attacker has the additional advantage that the DS^2 encoding is error correcting. Hence for each combination it could execute the error correction procedure and look for the correct combination of at least $h+\lfloor t/2 \rfloor$ moduli and residues. Note that applying the error correcting procedure would increase the workaround factor required by the attacker to break the confidentiality, however in the following, we do not take into consideration such a factor. Hence, the resulting analysis is a lower bound on the efforts required to the attacker to break the confidentiality of the scheme for this type of attack. Note that, taking into account the possibility for the attacker to exploit the error correcting features of the *RRNS*, the number of combinations in Equation (5) should be divided by

$\Phi = \sum_{k=0,\lfloor t/2 \rfloor} \left(\binom{h+t}{k} \frac{(r-t+k)!}{(r-t)!} \right)$, where, for each k, the first factor accounts for all the possible combinations of k residues associated with the wrong moduli, and the second factor accounts for all the possible association of these residues with all the moduli which are not correctly assigned.

From Figure 2, which shows the value $C = Log_2 \frac{\Theta}{\Phi}$ for r=8, t=4, and $h \in [10,35]$ and the factorial of h, it is seen that the logarithm of the number of combinations C grows as the logarithm of $h!$.

As an alternative, the attacker may generate all the possible integers in the range $[0,M)$ and encode each integer using the set of available moduli, until it finds an integer X whose encoding produces a set of residues which includes the set $\{x_1,\ldots,x_{h+t}\}$ of the received residues. It should be observed that this condition is not sufficient to guarantee that X equals the original encoded information, however we disregard this possibility to the advantage of the attacker. It is immediate that, disregarding the complexity of encoding, this procedure has complexity O(M). This leads to a complexity of about O($2^{16(h+r)}$), which however, considering the encoding with the moduli library of Table 1, results less efficient than the technique discussed above.

In principle a record could be reconstructed also by a mobile which has somehow received at least $t<h$ residue digits. Since the correspondence of residue digits with the moduli is unknown to the malicious mobile, it should then consider all possible t-tuples of residues in association with all permutations of the h+r available moduli. Assuming that the correct association of a t-tuple of residues with the t moduli is somehow guessed, the decoding procedure only yield an integer which is congruent to the record. The record could be recovered by adding some (unknown) multiple of the product of some of the additional moduli.

Decoding the residues in this way leads to many legitimate numbers, and it may be very difficult to determine the right one. If the records are plain ASCII encoding, a clue may be provided by the fact that most legitimate numbers do not correspond to ASCII encoding, on the other hand, to save wireless bandwidth and mobiles storage, the encoding and dispersal of the file is most likely preceded by file compression and hence DS^2 generally operates on binary files.

Integrity. We consider any attempt of a malicious mobile to modify a record. Note that the resilience to this type of attack is independent whether the modification is meaningful or not, that is, the attacker tries to modify the record in such a way that the resulting record has a different, meaningful information content, or the attacker randomly modifies the record, which is (wrongly) recognised as a correct one by the legitimate requester, despite its information content.

With the DS^2 encoding, malicious corruptions of residues can be recovered if h+t ($t \leq r$) residues can be read and no more than $\lfloor t/2 \rfloor$ residues are corrupted, and can be detected if the number of corrupted residues does not exceed t. This feature of DS^2 is an improvement with respect to the behaviour of recent standard protocols, which are subject to this type of threat, as in [21].

Availability. For every read operation, the disruption of up to r residues of a record do not compromise the capability recovering the record. Moreover, if the residues are routed through paths which have minimal intersection points (the ideal situation would be to have disjoint paths), the probability of DS^2 to be resilient to a DoS attack enhances dramatically. Indeed, for the attacker to be successful, it should take control of at least r+1 different specific nodes, each one being on one of the disjoint paths.

As a final remark, we note there is a number of little security flaws to which DS^2 is exposed, but which can be easily circumvented. For example, if the encoding produces a residue containing the value 65535, it is immediate that this residue must correspond to modulo m_0=65536. Also, the encoding of a record whose decimal content is smaller than any module of the *RRNS* yields residues whose content is the same as the original record. These little flaws can be easily prevented, for example by selecting a random constant (to be kept secretly in the file descriptor) to be added modulo 65536 to each residue after the encoding.

7 Comparison of DS^2 with Information Dispersal Algorithm (IDA)

We now compare DS^2 with the Rabin's Information Dispersal Algorithm (IDA) [10]. To this purpose we briefly review IDA.

Let us consider a file f composed by N characters $b_1,\ldots,b_N$ where each b_i is in the range $[0,B)$, and let p be a prime with $p>B$. Typical values of B and p are B=256 and p=257. As in DS^2 the file encoding produces $h+r$ fragments to be dispersed in the network, and the file content can be reconstructed from any h fragments.

Let us assume for the sake of simplicity that $N=(h+r)h$. To the purpose of encoding, IDA partitions the file in $h+r$ records $S_1,\ldots,S_{h+r}$, each of h characters, that is, $S_i=(b_{h(i-1)+1},\ldots,b_{ih})$ for each $i\in[1,h+r]$. Then it selects a $h+r$-tuple $(a_1,\ldots,a_{h+r})$ of $h+r$ randomly chosen vectors such that $a_i=(a_{i1},\ldots,a_{ih})$, $a_{i,j}\in[0,p)$, and any subset of h vectors is linearly independent.

The encoding produces $h+r$ fragments $F_1,\ldots,F_{h+r}$ each of size $h+r$ characters, where $F_i=(c_{i1},\ldots,c_{ih+r})$ and $c_{ik}=(a_i{\times}S_k)$ mod p for each $i,k\in[1,h+r]$.

Note that the characters of the fragments $F_1,\ldots,F_{h+r}$ are in the range $[0,p)$, while the characters of the file are in the range $[0,B)$ with $B<p$. This in general leads to a wastage of at least 1 bit per character, however with a simple technique [10] this overhead can be avoided.

The decoding procedure of IDA requires at least h fragments of the encoded file. Assume that $F_1,\ldots,F_h$, are available, the file can be reconstructed as follows. Let **A** be the $h{\times}h$ matrix whose ith row is a_i. Then record S_k is given by:

$$S_k = \mathbf{A}^{-1}\begin{bmatrix} c_{11} \\ \ldots \\ c_{h1} \end{bmatrix} \tag{6}$$

Disregarding the memory overhead to store the vectors, the IDA's encoding is optimal, since the overhead for an r-erasure tolerance encoding of a file is $(h+r)/h$. Letting a record size of $b = \lfloor \log_2 M \rfloor$ in DS^2, the code efficiency of IDA and DS^2 is almost the same.

Encoding/Decoding Efficiency. It is immediate that for IDA, the time complexity of the encoding of a file of size N=(h+r)h is O(N(h+r)) plus the cost of generating the set of vectors, while the complexity of decoding is O(hN) plus the cost of inverting matrix **A**. Although for sufficiently large files the costs involved by the generations of the vectors and the matrix inverse becomes negligible, Rabin shown efficient techniques for generating matrices which are invertible in time O(h^2).

Considering a file of size 8N bits, with N=(h+r)h, the complexity of encoding the file with IDA is O(h(h+r)^2). In the encoding of the same file with DS^2, the file is preliminary partitioned into 8N/b records with b≈16h. Hence the encoding with DS^2 has complexity O(h(h+r)N/h)= O(h(h+r)^2), which equals the complexity of the encoding of IDA. The complexity of the file decoding with IDA is O(h^2(h+r)), while the decoding with DS^2 is, in general, O((h+r)^3) which is slightly worse than the decoding complexity of IDA. However, if only erasures are considered, the decoding with DS^2 is O(h^2(h+r)), which equals the complexity of decoding with IDA. Table 2 summarizes the different encoding/decoding running time for the two algorithms.

Table 2. Encoding/Decoding Efficiency.

	DS^2	IDA
Encoding	O(h(h+r)^2)	O(h(h+r)^2)
Decoding h+r residues	O((h+r)^3)	-
Decoding (only erasures)	O(h^2(h+r))	O(h^2(h+r))

Confidentiality. In IDA, the recovery of a file requires knowledge of (a) the vectors of the matrix **A**; (b) the relative positions of the vectors of matrix **A**; (b) the knowledge of the nodes where the fragments are located; (c) the knowledge of the correspondence of each fragment with the appropriate position in the vector to be multiplied for **A**$^{-1}$, as in Equation (6). It is straightforward to map the requirements for IDA and those for DS^2. Indeed, in DS^2 what is needed is (a) the knowledge of the values of the moduli; (b) the knowledge of the nodes where the residues are located; (c) the knowledge of the correspondence of each residue with the appropriate module.

If the set of moduli is kept secret in DS^2 and the vectors composing matrix **A** is kept secret in IDA, the file sharing requires to transmit to the intended receivers such data. In this case the overhead of transmission of the (h+r) moduli requires at most 16(h+r) bits, which is lower than the overhead required to transmit the matrix **A** (which accounts for h(h+r)Log$_2$ p bits). However, in the following we will assume that the set of moduli is publicly known, as well as the vectors of the matrix **A** (but not their relative position). Under this hypothesis assume that that h+t fragments are correctly recovered. Than, the security of IDA relies on the workload required to the attacker to correctly identify the correct position of vectors to obtain matrix **A**, and bound the fragment with the appropriate position of the vector to be multiplied for **A**$^{-1}$, as in Equation (6). The security analysis for these points results in Equation (5).

However, in IDA also the access to less than *h* fragments may provide some information about the file. For this reason, to enforce confidentiality, the file could be encrypted before being encoded and dispersed by IDA, thus introducing additional overhead.

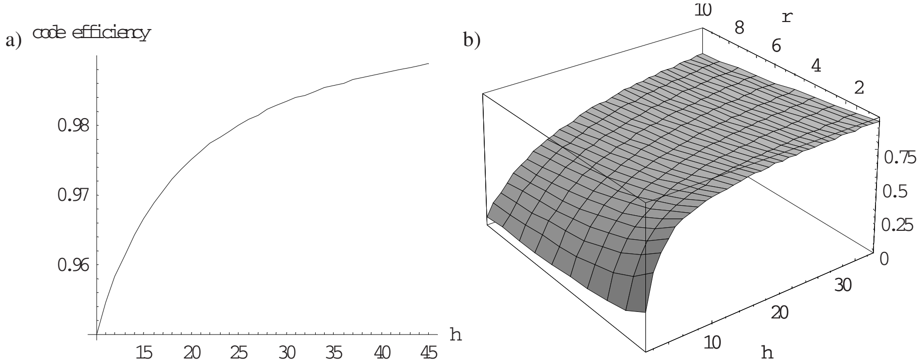

Fig. 1. Code efficiency for: (a) $h \in [10,45]$ and record size $b' = 16h - 8$, and (b) $r \in [1,10]$, $h \in [1,35]$, and record size $b' = 16h - 8$.

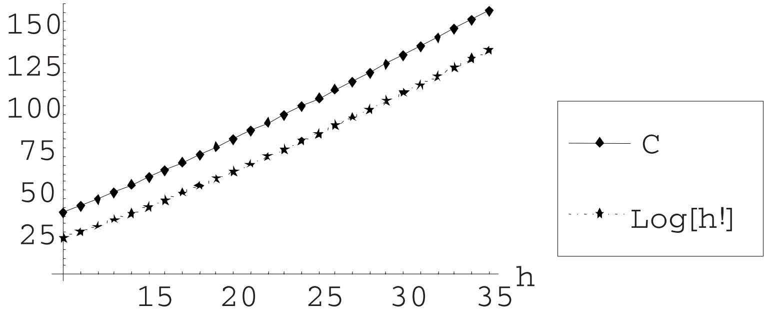

Fig. 2. Number of combinations *C* to be analyzed in case of brute force attack compared with $h!$ ($r=8$, $t=4$, and $h \in [10,35]$).

On the other hand, in DS^2 no information can be easily recovered from any subset of residues if the association between moduli and residues is not known, and a brute force attack would require a number of operations which grows as the factorial of *h* (for values of $h \le 50$).

Availability and Integrity. DS^2 employs an *RRNS*-based encoding which features error correcting capabilities, such that corruptions of less than $r/2$ residues can be recovered, and corruptions of up to $r/2$ residues can always be detected. Hence the availability and integrity requirements are easily met in DS^2.

As compared to DS^2, the IDA encoding is designed to cover only erasures. This means that, in case of malicious corruption of fragments, the original content of the file cannot be recovered. A solution could be to introduce fingerprints, that is, an indicator of whether the fragment has been altered or not, and hence discarded in the former case. However, the use of fingerprints introduces both computational and storage overhead to the encoding. Resorting to standard fingerprinting functions, like SHA-1, MD5 [20] has a computational complexity which is linear in the size of the block, but requires at least 128 bits per fragment. Note that without fingerprinting, both availability and integrity are at a stake. Moreover, only fingerprinting is not enough to prevent violation of the integrity requirement. Indeed, an attacker could replace the fragment F_i with an arbitrary one (say F_i'), compute the fingerprinting ($H(F_i')$) for this fragment and than send the tuple (F_i', $H(F_i')$) which will pass the integrity check. This would result in violation of the security requirements. This problem can be solved using keyed-fingerprinting [20], that is, the fingerprint is a function of two parameters, a key (shared only by legitimate users) and the message. Hence, the tuple (F_i', $H(k,F_i')$) can not be replaced, unless key k is known.

Both integrity and availability of IDA can be enforced resorting to keyed fingerprinting. However, this would result in an additional overhead in both computation and storage. While the computational overhead can be considered negligible, the storage requirement is not. Furthermore, the longer the fragment is, the more it is subject to possible errors during the transmission. Once a corrupted fragment reaches destination, it would not pass the integrity check, and hence will be discarded. Hence, to achieve the same degree of reliability and security as DS^2, IDA might require an higher degree of redundancy.

8 Conclusions

This paper discusses a dependable and secure data storage for mobile, wireless networks (DS^2) based on a peer-to-peer paradigm. DS^2 provides support to create and share files under a write-once model, and ensures at the same time data confidentiality and dependability by encoding files in a Redundant Residue Number System. More specifically files are partitioned into records and each record in encoded separately as (h+r)-tuples of data residues using h+r moduli. In turn, the residues are distributed among the mobiles in the network. Dependability is ensured since data can be reconstructed in the presence of up to $s \le r$ residue erasures, combined with up to $\left\lfloor \frac{r-s}{2} \right\rfloor$ corrupted residues, and data confidentiality is ensured since recovering the original information requires knowledge of the correspondence between moduli and residues. The achievable degrees of dependability and security are determined by the choice of the *RRNS* (that is, of the set of moduli).

The paper analyzes the code efficiency of DS^2 using a library of moduli which allows for efficient encoding and decoding procedures based on single precision arithmetic, and discusses the security issues. The comparison of DS^2 with IDA shows that the two approaches have almost the same performance in terms of code efficiency and complexity, even though DS^2 provides richer security features than IDA, exploiting the *RRNS* codes. To achieve the same level of security of DS^2, IDA requires additional overhead in both computation and storage resources.

References

1. D. B. Johnson and D. A. Maltz: Dynamic source routing in ad hoc wireless networks. Mobile Computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153–181. Kluwer Academic Publishers (1996)
2. C. E. Perkins and E. M. Royer:Ad-hoc on-demand distance vector routing. Proc. IEEE 2nd Workshop on Mobile Computing Systems and Applications, New Orleans, LA (1999) 90-100
3. L.M. Feeney and M. Nilsson : Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. Proc. IEEE 20th INFOCOM, Vol. 3, Anchorage AK (2001) 1548 –1557
4. C. A. Thekkath, T. Mann, and E. K. Lee : Frangipani: A Scalable Distributed File System. Proc. ACM 17th Symposium on Operating Systems Principles (1997) 224-237
5. Satyanarayanan, M.; Kistler, J.J.; Kumar, P.; Okasaki, M.E.; Siegel, E.H.; Steere : Coda: a highly available file system for a distributed workstation environment. IEEE Transactions on Computers, Vol.39 (4) (1990) 447–459
6. T. E. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang: Serverless Network File System. Proc. ACM 15th Symposium on Operating System Principles, Copper Mountain Resort, Colorado (1995) 109–126
7. J. C. Fabre, Y. Deswarte, and B. Randell. "Designing secure and reliable applications using fragmentation-redundancy-scattering: an object-oriented approach. Proc. 1st European Dependable Computing Conference, Berlin, Germany (1994) 21–38
8. Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos: A Prototype Implementation of Archival Intermemory. Proc. ACM 4th Conference on Digital libraries, Berkeley, CA (1999) 28–37
9. J. Kubiatowicz et. Al.: OceanStore: An Architecture for Global-Scale Persistent Storage. Proc. ACM 9th Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA (2000) 190–201
10. M. O. Rabin : Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. Journal of the ACM, Vol.36, (2) (1989) 335–348
11. L. Rizzo : Effective Erasure Codes for Reliable Computer Communication Protocols. ACM Computer Communication Review, Vol. 27 (2) (1997) 24–36
12. S. Chessa and P. Maestrini : Dependable and Secure Data Storage and Retrieval in Mobile, Wireless Networks. Proc. IEEE DSN 2003, International Conference on Dependable System and Networks, San Francisco, USA (2003)
13. F. Barsi and P. Maestrini : Error Correcting Properties of Redundant Residue Number Systems. IEEE Transactions on Computers, Vol. C-22 (3) (1973) 307–315
14. D. Mandelbaum : Error Correction in Residue Arithmetic. IEEE Transactions on Computers, Vol. C-21 (6) (1972) 538–545
15. N. S. Szabo and R. I. Tanaka, Residue Arithmetic and its Applications to Computer Technology, Mc Graw-Hill, New York (1967)
16. D. Mandelbaum : On a Class of Arithmetic Codes and Decoding Algorithm. IEEE Transactions on Information Theory, Vol. IT-21 (1976) 85–88

17. F. Barsi and P. Maestrini: Improved Decoding Algorithms for Arithmetic Residues Codes. IEEE Transactions on Information Theory, Vol. IT-24 (1978) 640–643
18. J. D. Sun and H. Krishna: A coding theory approach to error control in redundant residue number systems. II. Multiple Error detection and correction. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 39 (1) (1992) 18–34
19. M. Soderstrand, W. K. Jenkins, G. A. Jullien and F. J. Taylor, Residue Number Arithmetic: Modern Applications in Digital Signal Processing, IEEE Press,NY (1986)
20. M. Bellare, R. Canetti, and H. Krawczyk: Keying hash functions for message authentication. in Advances in Cryptology – Crypto 96 Proceedings, LNCS Vol. 1109, N. Koblitz ed, Springer-Verlag (1996)
21. Nikita Borisov, Ian Goldberg, David Wagner: Intercepting mobile communications: the insecurity of 802.11. MOBICOM 2001, Rome, Italy (2001) 180–189