

Unlinkability in Anonymous Publication Systems

Christian Boesgaard

pink@diku.dk

Department of Computer Science

University of Copenhagen

Denmark

January 26, 2004

Abstract

Anonymous publication systems must be censorship resistant and provide high availability of documents. If documents are stored as pieces on different servers, which can be linked to illegal documents, an adversary might be able to use this link to exercise censorship. Secret sharing can be used to create and store pieces without links to documents but cannot easily be combined with space efficient redundant storage schemes to support availability. This paper discuss the property needed to resist attacks based on links between documents and pieces. Furthermore, a storage scheme is presented that fulfills the property and provides space efficient redundant storage.

1 Introduction

The primary goal of an anonymous publication system (APS) is to ensure that published documents are protected against censorship. Documents are stored on a number of servers in an APS as a set of one or more pieces. A problem in existing systems is that there is a link between the pieces stored in the system and documents that can be retrieved from the system. An adversary might be able to use this link to provide censorship by mounting a legal attack, the adversary could use the law to force pieces to be removed from a server because it could be demonstrated that the pieces originated from an illegal document. Even in a system where servers are anonymous, an adversary might be able to search a large set of servers and be able to link pieces on the servers to specific documents, to order the removal of the pieces.

Secret sharing can be used to store documents in pieces, such that the pieces on the servers cannot be linked to specific documents if the pieces are stored together with any information about what document they can be used to recreate.

Another problem that must be solved in an APS is to make documents available even if some of the servers fail. Replication of content is a simple, but space inefficient solution to this problem. Space efficient solutions include

Rabin's information dispersal algorithm (IDA) [5], which can create a number of pieces from a document, where any subset with a specified number of pieces can be used to recreate the document.

It is possible to combine secret sharing and replication of pieces to provide unlinkability and increase availability, but it is difficult to combine secret sharing and IDA in a space efficient scheme. This is the problem I tackle in this paper. The contributions of this paper are: (1) a definition of the property required in an APS to resist censorship from adversaries using the link between documents and pieces on servers, and (2) a storage scheme based on IDA that has this property and supports space efficient redundant storage.

2 Unlinkability

In most APSs, for example Freenet [2] or Free Haven [3], an encrypted version of a document is stored in on one or more servers and the operator of a server will not easily be able to find the key for the pieces, the operator stores. This makes it possible for a server operator to plausibly deny that she has any knowledge of the documents her pieces can be used to recreate. In such an approach, there is an explicit link between pieces on a server and a document. In Freenet, documents are stored on a single server; in Free Haven a document is divided into a number of pieces using IDA [5], which are stored together with information that links the pieces together.¹ The explicit link can be used by adversaries to mount legal attacks:

- In APSs where storage servers are not anonymous, an adversary who finds information on how to recreate an illegal document can mount a legal attack against the storage servers that store pieces that originates from the document.
- In APSs with anonymous storage servers, an adversary might get access to search a set of servers and find out what documents are connected to which storage servers and use this as a basis for legal attacks if any illegal documents are found.

To provide resistance against such attacks, it should be impossible for an adversary to demonstrate a connection between a piece on a server and an illegal document. Let us define this property, using **share** to denote pieces,² as *unlinkability*:

Unlinkability is a state where a share s can be used to recreate more than one document.

In an APS providing unlinkability, an adversary would however still be able to demonstrate a connection between a document and a share, as the set of documents a share could be linked to might be rather small. Let us define a stronger property without any connection between shares and a finite set of documents:

¹ It is however not possible to find the other pieces given one piece.

² *Share* is used instead of *piece*, as this is the notion normally used in relation to secret sharing.

Perfect unlinkability is a state where a share s can be used to recreate any document.

If a share s , in an APS, originates from the DeCSS³ code. The APS provides perfect unlinkability if s can be used to recreate any document. This makes it impossible (at least in theory) to demonstrate a link between s and the DeCSS code.

Perfect unlinkability is a weaker property than perfect secret sharing, as it does not require that nothing can be known about a set of shares relation to a document. Perfect unlinkability is a variant of Shannon's definition of perfect secrecy [7] that basically states that given a ciphertext, nothing can be learned about the plaintext it represents.

Unlinkability makes it harder for an adversary to make attacks against an APS by targeting servers in the system using the shares they store, as it is harder to show a connection between the stored shares and specific documents. This might also strengthen a server operator's ability to deny knowledge of the shares on her server compared to storage schemes with explicit links between shares and documents, such as encryption keys in Freenet [2] or identifiers in Free Haven [3].

In theory perfect unlinkability makes it impossible for adversaries to connect illegal documents to shares in an APS. However, a powerful adversary might be able to collect all shares that ever existed in an APS and try all combinations to find out what documents could be created from what shares, this would probably convince a court.⁴

3 Providing Perfect Unlinkability

In the following I discuss how a storage scheme can provide unlinkability. The following assumes an APS with a number of servers where documents are stored on a set of servers in fixed sized pieces. I use **block** to refer to part of a document and **recipe** to refer to the information a user needs to have to be able to recreate a document.

Secret sharing [4, Section 12.7], such as Shamir's threshold scheme [6] or simple schemes based on exclusive-OR [1], can be used in an APS to store documents as shares on different storage servers. A storage scheme based on perfect secret sharing can make it impossible to link shares on storage servers to any specific documents as perfect secret sharing provides perfect unlinkability.

The problem with existing perfect secret sharing schemes is that it is inefficient to provide redundancy with these, because the shares is of the same size as the document that is secret shared. Furthermore, if a scheme like Shamir's is used, there will be additional information that have to be stored in a redundant way.

An alternative approach is to use an information dispersal algorithm, such as IDA [5]. IDA provides space efficient redundant storage: if a block requires space equal to m shares to be recreated, IDA can be used to create n shares, where every subset of m different shares can be used to recreate the block.

³ DeCSS Central, <http://web.lemuria.org/DeCSS/>

⁴ A partial solution to this problem is to ensure that all the shares in an APS can be used to create a set of documents.

However, IDA does not in itself provide perfect unlinkability. I will begin with an explanation of IDA and then present a scheme based on IDA that provides perfect unlinkability.

3.1 IDA

IDA is used to divide a block F of size L into n shares of size $s = L/m$, where any subset of m distinct pieces can be used to recreate the block. I will use **share** to denote such a piece.

n is chosen such that $n = m + k$, where k is the number of shares that are redundant.

IDA is based on computations in a finite field \mathbb{Z}_p (where p is a prime). The block is treated as a list of numbers, b_1, \dots, b_k , where $b_i < p$.

F is divided into pieces each consisting of m numbers: F_1, \dots, F_s ($F = (b_1, \dots, b_m); (b_{m+1}, \dots, b_{2m}); \dots$).

In IDA an $n \times m$ dispersal matrix D is used to create the shares. The matrix must be constructed such that it contains n rows that are linearly independent. Another $m \times s$ matrix \mathcal{F} is created with F_1, \dots, F_s as the columns. The two matrices are multiplied to get an $n \times s$ matrix \mathcal{R} where each row corresponds to a share.

To recreate F from a set of shares, the shares as well as the corresponding row in D are needed. The rows from D is used to create a matrix D' , the shares to create \mathcal{R}' matrix. When D'^{-1} is multiplied with \mathcal{R}' , the result is a matrix containing F . The use of IDA is illustrated with an example in Appendix A.

There is a space overhead on IDA, because a number of rows from D is needed to recreate \mathcal{F} . For small values of n and m the matrix can be stored together with the recipe. For large values of n or m the rows can be stored together with the shares.

3.1.1 IDA and Perfect Unlinkability

IDA does not provide perfect unlinkability if used as described previously. If the rows from D is saved together with the shares r from \mathcal{R} , the maximum size of the original block is given by the m elements in the row from D . This makes perfect unlinkability impossible because the size of the block limits the possible blocks.

If the rows from D are not saved together with the shares, D must be significant smaller than \mathcal{F} (because D is needed to recreate \mathcal{F}).⁵ That is $n \cdot m \ll m \cdot s \Rightarrow n \ll s$. This scheme cannot provide perfect unlinkability, because given an arbitrary share (a row in \mathcal{R}) it would require that a row in D could be created, which mapped an arbitrary \mathcal{F} into the share. But as a row in \mathcal{R} contains s elements and a row in D m contains elements, this is impossible for $m \ll s$, as the row in D cannot be used to create all the possible mappings. Furthermore, note that if the rows from D are not saved together with the shares, then nothing can be learned about the size of F .

⁵ If D is larger or around the same size of \mathcal{F} , it does not make sense to use the APS as it would be necessary to distribute D that is about the same size or larger than F itself.

3.2 Providing Perfect Unlinkability with IDA

A storage scheme based on IDA can provide perfect unlinkability if the rows from \mathcal{D} are not saved together with the shares and if s random elements are padded to F before IDA is applied (corresponding to an extra row in \mathcal{F}). The idea is that the requirements of the possible F 's that can be recreated is reduced because the possible \mathcal{F} s is limited by a set of s linear equations in m variables with coefficients from a row in \mathcal{D} that defines the s columns in \mathcal{F} and variables that define a column in \mathcal{R} . Because it is linear equations and one row can be given any value, it is possible to recreate any F , representing any block by defining all rows in \mathcal{F} except the last one that is calculated.

In short: given any F and any share S , it is possible to create a \mathcal{D} , a \mathcal{F} containing F , and a \mathcal{R} containing S .

In most APSs it is required that shares on storage servers are of a fixed size. This is a problem with this scheme, as fixed size shares means that m and n changes with the size of documents. For small documents \mathcal{D} can be saved together with the recipe, but for larger documents this is not feasible. One alternative is to save the rows from \mathcal{D} as part of the shares together with the rows from \mathcal{R} , this is however also a problem as the size of the rows in \mathcal{D} is the size of the variable m , which means the share will not be a fixed size.

A possible solution is to save \mathcal{D} for large documents as either a replicated share or a "document" using IDA. The disadvantage of this approach is that it requires more space to provide the same amount of redundancy as solutions saving the rows from \mathcal{D} either as part of the shares or in the recipe. For \mathcal{D} larger than a share, \mathcal{D} could be saved in multiple shares using IDA, with a small \mathcal{D}' that could be saved in the recipe for the document.

I use the following to describe the properties of this scheme: a document of size s_d is to be stored with a redundancy factor r as shares of size s_s , which means that $n = s_d/s_s$ shares equals the size of a document. The space needed to store a document is $r \cdot (s_d + k)$ (excluding space for \mathcal{D}), where k is the small amount extra space needed to store an extra row in \mathcal{F} . And a user must retrieve $2 \cdot (s_d + k)$ data to recreate a document. The probability of the document being available is $a = \sum_{i=n}^{r \cdot n} \binom{r \cdot n}{n} p^i (1-p)^{r \cdot n - i}$, where p is the probability that a share is available.

3.3 Discussion

The scheme I have presented can be compared to the use of a secret sharing and combinations of secret sharing and IDA. To use a storage scheme based on only secret sharing scheme, for example Shamir's threshold scheme [6], it would be necessary for the storage scheme to divide a document into pieces the size of a share before the secret sharing. These pieces would have to be secret shared independently. The effect of this is that the availability of the document would fall exponentially with the document size. A combination of secret sharing and IDA can provide better availability. In the APS YÅPS [1] a combination of secret sharing based on exclusive-OR and IDA is used to create shares: a document is first secret shared using exclusive-OR with a random block b_r the size of the document, which creates another block b_s of the same size. Each block is then dispersed into shares using IDA. This scheme provides perfect unlinkability and space needed is $2(r \cdot s_d)$ and a user must retrieve $2 \cdot s_d$

data to recreate a document. The probability of the document being available is equal to the probability that both b_r and b_s is available, which is equal to a^2 (using the definition of a from the last section). I have compared this scheme to the scheme presented in this paper in Table 1, ignoring some details.⁶

Scheme	space requirement	retrieval	availability
IDA, \mathcal{D} in recipe	S_1	S_2	a
IDA, \mathcal{D} in a share	$S_1 + r \cdot s_s$	$S_2 + s_s$	$a \cdot (1 - p)^r$
YÅPS	$2S_1$	$2S_2$	a^2

Table 1 Overview of storage schemes where comparisons is made to the IDA solutions. S_1 and S_2 is introduced as a base for comparison, the rest of the variables used is defined in the previous Sections.

The IDA-only schemes use around half the space as the YÅPS scheme. This also holds for the data a user needs to retrieve to recreate a document (or send to publish a document). For small documents the IDA scheme provides better availability at the same space usage, whereas for larger documents it is comparable to the availability in the YÅPS scheme.⁷

4 Related Work

The notion of a hidden link between pieces on servers and documents as a protection for server operators is discussed in relation to the Free Haven project [3].

In my master’s thesis [1], I introduced unlinkability as a feature called *strong deniability* and presented the APS YÅPS that supports a storage scheme using secret sharing to provide unlinkability and IDA to provide redundancy. That scheme does not support entanglement.

An alternative countermeasure to legal attacks was introduced in *Tangler*[8] (a centralized APS). *Tangler* provided *entanglement* of documents by making it possible to reuse shares to store different documents. Furthermore, the idea of *Tangler* has been discussed in relation to plausible deniability to protect the users, because it removes the relation between retrieving a specific share and a specific document.⁸ Compared to unlinkability, entanglement can be seen as a means to demonstrate to a court that a share is unlinkable with a specific document. The scheme presented in this paper can provide limited entanglement.⁹

⁶ Simplifications include the exclusion of k for the IDA schemes and IDA where \mathcal{D} requires more than one share to be stored.

⁷ The availability of the IDA scheme can be adjusted at a small trade off with space.

⁸ Roger Dingledine of the Free Haven project, discusses entanglement on the development mailing list, April 15 2001, <http://archives.seul.org/freehaven/dev/Apr-2001/msg00007.html> (January 2004).

⁹ The scheme supports limited entanglement because a random share can be reused in the generation of the \mathcal{D} when a block is to be shared.

5 Conclusion

In this paper, I have defined the property unlinkability that an APS must fulfill to be resistant to legal attacks. I have also presented a storage scheme based on IDA that provides perfect unlinkability and space efficient redundant storage of documents in fixed sized shares for APS. In short, the presented scheme uses around half the space of existing solutions based on a combination of secret sharing and IDA.

References

- [1] Christian Boesgaard. YÅPS: Yet another anonymous publication system. Master's thesis, Department of Computer Science, University of Copenhagen, December 2003. Report Number 03-03-28, <http://zibet.net/haven/>.
- [2] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer-Verlag, Heidelberg, 2001. URL citeseer.nj.nec.com/clarke00freenet.html.
- [3] Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000*, volume 2009 of *Lecture Notes in Computer Science*, pages 67–95. Springer-Verlag, Heidelberg, 2001. URL citeseer.nj.nec.com/dingledine00free.html.
- [4] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997. ISBN 0-8493-8523-7.
- [5] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989. ISSN 0004-5411.
- [6] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11): 612–613, November 1979. ISSN 0001-0782.
- [7] Claude Elwood Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949. <http://www.cs.ucla.edu/~jkgong/research/security/shannon.html>.
- [8] Marc Waldman and David Mazi. Tangler: a censorship-resistant publishing system based on document entanglements. In *ACM Conference on Computer and Communications Security*, pages 126–135, 2001. URL citeseer.nj.nec.com/waldman01tangler.html.

A IDA Example

IDA Example: $F = (4, 2)$ is to be dispersed using \mathbb{Z}_{257} . F will be dispersed into 4 shares such that only two pieces are needed to recreate F . That is, $L = 2$, $m = 2$, $n = 4$, and $s = 1$, which means the block is treated as one piece.

First a \mathcal{D} is chosen:

$$\mathcal{D} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}$$

\mathcal{D} is then multiplied with \mathcal{F} to get \mathcal{R} :

$$\mathcal{R} = \mathcal{D} \times \mathcal{F} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \times \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 8 \\ 20 \\ 32 \\ 44 \end{bmatrix}$$

It is then possible to recreate F from two of the four pairs: $p_1 = ([1 \ 2][8])$, $p_2 = ([3 \ 4][20])$, $p_3 = ([5 \ 6][32])$, $p_4 = ([7 \ 8][44])$. In the following F is recreated from p_1 and p_2 , using these to construct \mathcal{D}' and \mathcal{R}' .

$$\mathcal{F} = \mathcal{D}'^{-1} \times \mathcal{R}' = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \times \begin{bmatrix} 8 \\ 20 \end{bmatrix} = \begin{bmatrix} 255 & 1 \\ 130 & 128 \end{bmatrix} \times \begin{bmatrix} 8 \\ 20 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$