Secure Processing Offload in Recombining Media Segments for Mobile Access

Thijs Veugen, Hans Stokking
TNO, Technical Sciences
P.O. Box 5050, 2600 GB, Delft, The Netherlands
{thijs.veugen, hans.stokking}@tno.nl
Thijs Veugen is also affiliated with the
Information Security and Privacy Lab of Delft University of Technology

Abstract

We develop an architecture for a federation of home gateways. We distinguish inner and outer circle friends and describe a mechanism for social-aware backup of content and sharing with friends. To facilitate remote access, content can be segmented and redundantly stored at gateways of friends. We develop innovative solutions that allow secure and redundant storage of segments with inner and outer circle friends. They enable decoding of segments in the encrypted domain without the need of sharing a decryption key with friends, which relieves the computational effort of mobile devices accessing the content. We combine Vandermonde-Reed-Solomon codes with homomorphic encryption to three different solutions and describe their differences in complexity.

I. INTRODUCTION

Nowadays, people create, share and consume content in a multitude of ways. Using their mobile devices, people take photos and shoot videos, and post them directly online to share with their friends, family and often with larger groups in their social network.

The FP7 FIGARO project focusses, among other things, on secure distributed backup, sharing and remote access to content, using people's social network. To enable this, the project has created an architecture of federated home gateways of the users. This federation allows a user's home gateway to make backups at the gateways of friends and family. These backups can be 'social' backups, to be used for the purposes of sharing the content with friends and enabling access to content while on the move [1][2].

If video content is of high quality, the bandwidth demands for streaming this content will be (much) higher than the average uplink of a single home gateway. Therefore, for the purpose of remote access, the backup of the (video) content is segmented into small parts. This allows for streaming of high-quality (HD) video, using the uplink bandwidth of multiple friends' gateways at the same time. Each gateway can then stream certain segments, to the extent the uplink bandwidth permits.

This paper focusses on secure remote access. We distinguish between two groups of friends, the so-called 'inner circle' and 'outer circle' friends. Inner circle friends are friends whom you completely trust, while outer circle friends are from the larger group in your social network, whom you do not trust completely.

We use Vandermonde-Reed-Solomon codes [4] to ensure the back-up can be retrieved as long as a limited number of home gateways is online. This is a common technique in storage technology also known as Redundant Array of Independent Disks (RAID). To avoid leakage of content towards outer circle friends we use homomorphic encryption [5]. The homomorphic property enables outer circle friends to decode segments in the encrypted domain, thereby reducing the workload of the downloading client to decryption only. The application of additively homomorphic encryption, but also of multiplicatively homomorphic encryption is demonstrated.

To further reduce the workload of the downloading (mobile) client, a solution based on symmetric encryption is described which accelerates the client's decryption effort. As with all our solutions, to reduce security risks we don't allow inner circle friends to learn the client's private key.

In Section II we describe our new architecture for social-aware backup and sharing of content. In Section III the security measures are described and analyzed, and compared with existing solutions. We end with the conclusions in Section IV.

II. ARCHITECTURE

FIGARO proposes an evolvable future Internet architecture based on gateway-oriented federation of residential networks. The residential gateway has a central role in the FIGARO vision of the future Internet. It interconnects the residential network with the Internet and is responsible for aggregating a multitude of devices and services within the residential network. In FIGARO, residential gateways undertake the federator role, internally as well as externally. Figure 1 shows residential networks connected at the edge of the Internet and illustrates a simplified view including the two types of residential network federations. The upper part illustrates external federation interconnecting multiple gateways to form a cooperative overlay across residential networks. This federation enables further collaboration to offer added value in terms of, for example, access and sharing of content, storage and network capacity. The right-part of the figure shows the internal federation within a residential network.

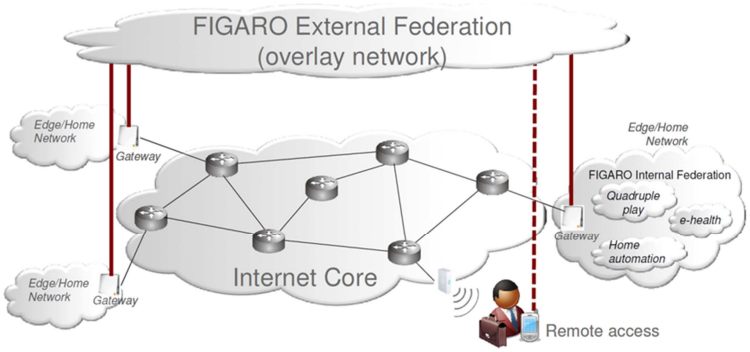

Figure 1: Overview of the FIGARO environment

A. *Social-aware backup and sharing*

Backup provides durability through duplication, while caching raises availability of content. The mechanisms of these functions can be combined to establish network efficiency when both doing backups and sharing content in a group of close friends. E.g. when backing up your family pictures, you can create a backup at your families gateways and at the same time share the content with your family. The content is then transmitted over the network only once, while serving the two purposes of both backup and sharing, including mobile access (see next subsection). This backup is comparable to work presented in [3]. Social backup is different in the sense that it is based on close relationships, e.g. family and good friends. We envision having offline contact with these relations, and agreeing on the use of the others' bandwidth and storage. This usage does not have to be reciprocal. A main insight here is that people you trust and know, are willing to help you, and this circumvents the idea of freeriding.

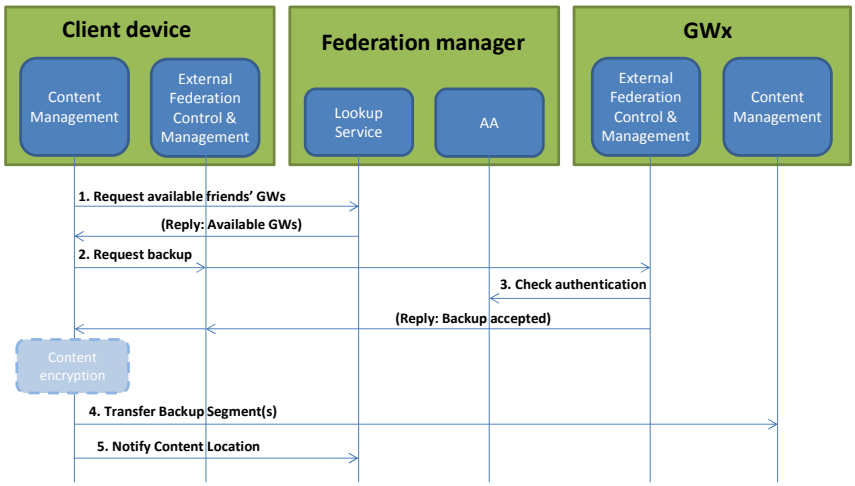

Figure 2: Social-aware backup performed by a client device to a GWx of a friend

The social-aware backup protocol in the FIGARO environment is shown in Figure 2. In this figure, a client device is shown making a backup of some (piece of) content at a gateway x (GWx) from some friend. In a first step, the client device requests a list of available friends' gateways from the Lookup Service. The Lookup Service is the part of the centrally located Federation Manager which amongst other things keeps track of all available gateways and of the users of those gateways. After receiving a list of available gateways and making a selection, the client device requests the backup from GWx, belonging to a friend. After GWx has authenticated the client device as belonging to a friend, the content is optionally encrypted (for explanation, see below). Then, the backup itself is performed. This backup is performed in segments to enable remote (mobile) access, see next subsection.

Whether the content is encrypted, depends on the role of the friend in the user's social network. We have divided the social graph of a user in two parts: inner circle friends and outer circle friends, as shown in Figure 3. Inner circle friends are the friends or family with whom you have a very strong connection, i.e. there is a high level of trust between you and them. Outer circle friends are the friends with whom you do connect online, but with whom the connection is less strong. For your inner circle friends, encryption of content is not performed, while for outer circle friends, it is.

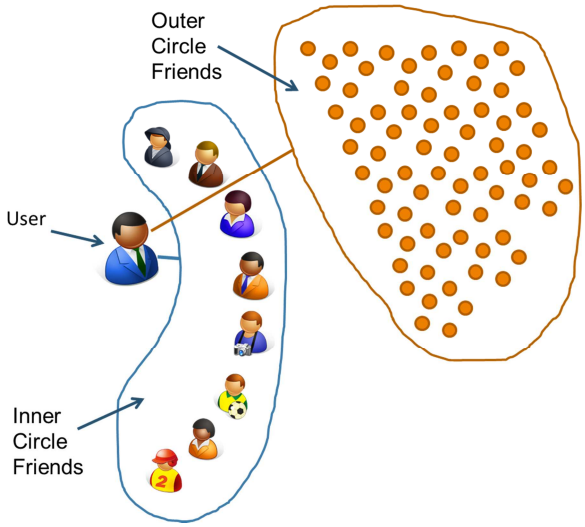

Figure 3: Inner- and outer circle friends

After backing up the content, the user can now share this content as well. If the content is shared with a user from a location at which a backup is placed, only an authorization step is necessary, as the content itself is already distributed. This is shown in Figure 4. The client device first requests the friend's GW location from the Lookup Service, and then sets the access rights for the other user in the AA module at the Federation Manager. These access rights are used in the mobile access scenario in the next subsection. Finally, a notification of the shared content is sent to the other GW, and optionally a key for access to the content.

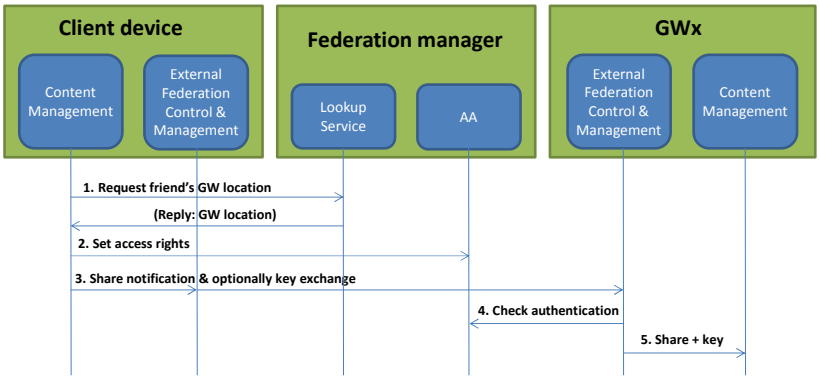

Figure 4: Sharing of previously backed-up content

Although the encryption itself does not need to be special for this type of backup and sharing, there are some new requirements to be met here. If different content is shared with different users, different keys need to be used in each occasion. While some content may be shared with a certain user, other content may not, while both content items can be backed up at that user's location. This requires different keys for different content items.

B. Mobile access

To enable remote access to content, either by the owner of the content or by users with whom the content is shared, content can be segmented. Reason for this is the limited upload network bandwidth most connections have. E.g. streaming of a high-quality video may require a 4 mbit/s connection, while many upload speeds are limited to e.g. 512 kbit/s or 1 mbit/s. By segmenting the content in small segments, and distributing these segments across various locations, even high-quality video becomes remotely accessible for streaming. This can thus be seen as a form of caching, included for free (i.e. without making additional copies in the network) in the backup and sharing scenario. For granting other users such access when sharing content, the FIGARO Federation Manager keeps track of these authorizations in the central AA function.

Figure 5 shows this mobile access scenario. The mobile device first has to discover the location of the backed-up content segments, and can then request these segments. Gateways containing these segments first have to check if the requesting device has authorization, and can then deliver the segments. If the users of the mobile device and the gateway(s) containing the segments are inner circle friends, then the segments will be delivered unencrypted. This saves the mobile device processing and thus precious battery life. If the users are outer circle friends, segments will be encrypted and it is up to the mobile device to decrypt the segments before having access to the content.

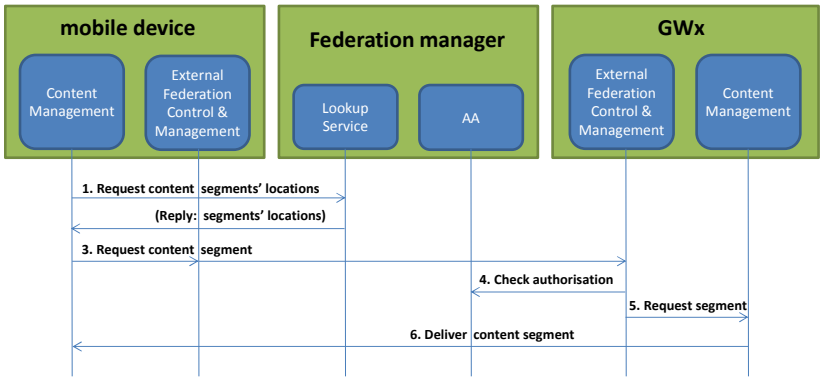

Figure 5: Remote access by a mobile device to backed-up content segments

III. SECURITY MEASURES

In order to realize both the "social-aware backup and sharing" as well as the "remote access" scenario in a secure way we need a couple of security measures to adequately protect content from outer circle and other unauthorized users, and assure its availability. A frequently used encryption system in such environments which offers different keys for each content is known as Convergent encryption [9], where the key is more or less a cryptographic hash of the content. Although this could be used in the "social-aware backup and sharing" scenario as described in Subsection II-A, the segmentation of content required in "remote access" asks for a more sophisticated solution.

To reduce content leakage we want to avoid storing a decryption key for each content (movie, document, etc.) at inner circle locations. The solution we propose uses Vandermonde-Reed-Solomon codes and homomorphic encryption. We first explain these two main concepts and then show how to combine these.

A. Vandermonde-Reed-Solomon codes

The first concept, also known as RAID, enables storing the content in a distributed way such that whenever a certain amount of users is available, the content can be retrieved. This measure guards the availability of content. More precisely, we store n chunks of the content, represented by some integer X, at various locations such that k chunks, $k < n$, are sufficient to restore the content.

Similar to [4], the key ingredient here is a *n* by *k* Vandermonde like matrix M that has the property that any *k* rows are linearly independent. This can be achieved [6] by defining the elements of M for each j, j = 1, …, *k*, as:

M_{ij} = δ_{ij} when i = 1, …, *k*;
j^{i-k-1} when i = *k*+1, …, *n*.

Here δ denotes the Kronecker delta. By using M, the content can be distributed and reconstructed as follows:

1. Divide content X into *k* integer valued segments X_1, …, X_k, each of size |X|/*k*.
2. Compute the *n* content chunks $(Y_1, \ldots, Y_n)^T = M * (X_1, \ldots, X_k)^T$ and store each chunk at a different location.
3. Suppose *k* chunks are available, represented by a subset S of {1, …, *n*} of size *k*. Let M_S be the matrix consisting of these *k* rows of M, and let Y_S be the vector consisting of these *k* chunks.
4. Since the *k* rows of M_S are linearly independent, this matrix can be inverted and the *k* segments can be reconstructed by $(X_1, \ldots, X_k)^T = M_S^{-1} * Y_S^T$.
5. The content X can be downloaded and combined from the *k* segments X_1, …, X_k.

Because the segments and the matrix elements are integers, the chunks Y_1, … ,Y_n will also be integers. However, because the matrix elements can be quite large, the size of the chunks might grow leading to storage size disadvantages. To overcome this problem, the computations in steps 2 and 4 could be performed in a finite field of size N such that N > |X|/*k* [6]. This condition assures correct reconstruction of the segments.

B. Homomorphic encryption

With respect to the confidentiality requirement, our solution should enable inner circle friends to retrieve the content but should avoid outer circle friends learning the content. For this purpose we introduce the second concept known as homomorphic encryption which is a form of encryption that allows a limited number of operations on encrypted data.

We distinguish between additively homomorphic encryption and multiplicatively homomorphic encryption and give examples for both systems. A well-known encryption system that is additively homomorphic is Paillier [7]. In Paillier, the encryption of x, $0 \leq x < N$, is $[x] = g^x * r^N \bmod N^2$, where g is a generator, r a fresh random value, and N a large RSA-like number consisting of the product of two large primes. Paillier is additively homomorphic because $[x] * [y] = [x + y] \bmod N^2$.

A well-known multiplicatively homomorphic encryption system is (unpadded) RSA [8]. In RSA, an encryption of x, $0 \leq x < N$, is $[x] = x^e \bmod N$ for some (public) integer *e* and number N that consists of the product of two large primes. RSA is multiplicatively homomorphic because [x] * [y] = [x * y] mod N. The encryption [x] is decrypted by raising it to the secret power *d*, where *d* is the multiplicative inverse of *e* mod φ(N).

In both encryption systems, one has to know the prime factors of N to be able to decrypt.

C. Secure and redundant storage with inner and outer circle friends

The concepts of redundant storage and homomorphic encryption can be nicely combined to a secure system that fulfills the requirements of both scenarios. We describe how content could be stored redundantly and securely, and retrieved by mobile devices, using additively homomorphic encryption. To reduce the workload of mobile devices retrieving the content from the network, we would like to have RAID decoding done by the inner and outer circle friends.

1. The content owner generates an instantiation of Paillier and broadcasts the public key to his inner circle friends.
2. To store content X, he divides content X into *k* integer valued segments X_1, … X_k, each of size |X|/*k*.
3. He computes the *n* chunks $(Y_1, \ldots, Y_n)^T = M * (X_1, \ldots, X_k)^T$ and stores each chunk at a different inner circle friend.
4. Inner circle friends are allowed to store copies of their chunks Y_i at outer circle friends, but only after encrypting it with the content owner's public key: $[Y_i]$.
5. Suppose *k* encrypted chunks are available from outer circle friends, represented by a subset S of {1, …, *n*} of size *k*. Let M_S be the matrix consisting of these *k* rows of M, and let Y_S be the vector consisting of these *k* chunks.

6. Since the *k* rows of M_S are linearly independent, this matrix can be inverted and the *k* segments can be reconstructed by $(X_1, \ldots, X_k)^T = M_S^{-1} * Y_S^T$. Because the chunks are now encrypted, reconstruction now has to be performed by outer circle friends in the encrypted domain:
   $[X_i] = \prod_{j \in S} [Y_j]^{a_{ij}} \bmod N^2$
   where the numbers a_{ij} are the elements of the matrix M_S^{-1}.
7. The content owner downloads the encrypted segments $[X_i]$ from the outer circle friends and decrypts them with his private key.
8. The content owner downloads and combines X from the *k* segments $X_1, \ldots, X_k$ and enjoys the content.

Note that the matrix M should be available to inner circle friends to be able to enjoy the shared content. This could be arranged by the Federation Management component as described in the previous section. Also outer circle friends that need to reconstruct the (encrypted) segments need to know the elements of the matrix M_S^{-1}. This could be achieved through inner circle friends or directly from the Federation Management component.

A disadvantage of Paillier is that encryption blows up the information size by a factor two, namely from N to N^2. Furthermore, decryption of segments by the mobile device of the content owner will be costly. This can be solved by a subtle change in encryption: instead of encrypting his chunk Y_i with the content owner's public key in step 4, the inner circle friend could encrypt his chunk with a self-generated symmetric key, e.g. by using Convergent encryption [9], and encrypt this symmetric key with the content owner's public key. This reduces storage size and simplifies the decryption by the mobile device, but on the other hand prevents the network from doing the RAID decoding for the user.

Alternatively, RSA could be used with avoids blowing up the size by encryption. However, because it is not additively but multiplicatively homomorphic, the encoding process has to be modified.

The chunks Y_j should be computed through exponentiation instead of multiplication (as in step 3):

$$Y_j = \prod_{i=1, \ldots, k} X_i^{M_{ji}} \bmod N$$

The reconstruction of segments in the encrypted domain (as in step 6) however is unmodified:

$$[X_i] = \prod_{j \in S} [Y_j]^{a_{ij}} \bmod N,$$

where the numbers a_{ij} are the elements of the matrix M_S^{-1}. Note that the relations in the plain domain are different: $X_i = \sum_{j \in S} a_{ij} * Y_j \bmod N$ with Paillier, but $X_i = \prod_{j \in S} Y_j^{a_{ij}} \bmod N$ with RSA. The computation of a matrix inverse requires the computation of multiplicative inverses. In Paillier these inverses have to computed in the field Z_N^*, but in RSA these are done in the field $Z_{\varphi(N)}$ which raises a problem because $\varphi(N)$ cannot be computed without knowing the factorization of N. However, by representing the numbers a_{ij} as rational numbers and choosing integer *c* large enough such that all numbers $c * a_{ij}$ are integers, outer circle friends will be able to reconstruct the segments $[X_i]^c$. The content owner is able to decrypt (in step 7) these segments to X_i by raising them to the power $d * c^{-1} \bmod \varphi(N)$.

D. Related work

A solution known from the sensor node domain [10][11] is to use symmetric encryption and secretly share the symmetric key with adjacent nodes, or in our domain with the inner circle friends. The disadvantage of this solution is that any k inner circle friends are able to retrieve the decryption key. An outsider somehow getting access to a sufficient number of key shares would also be able to retrieve the decryption key and thus obtain the content through (untrusted) outer circle friends. A second disadvantage of this solution is that content always has to be decrypted before usage, even by inner circle friends, which causes an undesirable computational effort.

To the authors' knowledge, our solution is unique in the sense that it enables RAID like decoding in the encrypted domain without the need of sharing the decryption key.

E. Complexities

In Subsection III-C we have described three different solutions that use RAID but avoid sharing of the decryption key by inner circle users. The computational and communication complexities of these three solutions are compared, both for the (mobile) client and for the network.

In the first solution, the client generates a Paillier key pair, and distributes the public key among inner circle friends. The public key is used by inner circle friends to encrypt their content chunks Y_i before storing it at outer circle friends. RAID decoding is done by the network in the encrypted domain.

The second solution is similar but uses RSA instead of Paillier which avoids data expansion by encryption but requires a different coding scheme. The main differences with the first solution are:

1. Different encryption and decryption algorithms (RSA vs. Paillier).
2. The size of an encryption is N^2 in Paillier but only N in RSA.
3. The computation of chunks (Y_1, …, Y_n) from segments (X_1, …, X_k) requires only multiplications with Paillier but exponentiations with RSA.
4. The computation of segments (X_1, …, X_k) from chunks Y_S requires exponentiations to the (rational) powers a_{ij} with Paillier but (integer) powers $c * a_{ij}$ with RSA. For small values of k and n the integer powers are smaller than the rational powers (of size N), but since a similar trick can be used with Paillier, this difference is negligible. However, because of Paillier's larger size of encryptions, each multiplication (and exponentiation) will require a larger effort than with RSA.
5. Because we use unpadded RSA, this crypto system is less secure than Paillier, which is semantically secure [7].

The third solution uses any public key encryption system (not necessarily homomorphic), and the public key is distributed among inner circle friends. Inner circle friends use Convergent encryption to encrypt their content chunks Y_i before storing it at outer circle friends. The symmetric key is also transmitted to outer circle friends after encryption with the public key. This solution increases the speed of encryption and decryption, but doesn't allow RAID decoding in the encrypted domain. The main differences with the first solution are:

1. Convergent encryption uses fast symmetric encryption and decryption algorithms. Public key encryption is only used for encrypting (small) symmetric keys.
2. Convergent encryption doesn't lead to a blowup in encryption size, but requires additional transmissions of the (encrypted) symmetric keys.
3. The computation of segments (X_1, …, X_k) from chunks Y_S requires exponentiations with Paillier, but is done in the plain domain with Convergent encryption and thus requires only multiplications.
4. The computation of segments (X_1, …, X_k) from chunks Y_S is done by the network with Paillier, but has to be performed by the client with Convergent encryption.

To conclude, the second, RSA-based solution seems to be preferable to the first, Paillier-based solution from a communication and storage complexity point of view. The disadvantages of the second solution being the higher computational complexity of the encoding scheme (to compute the chunks from the segments) and the weaker crypto system.
From a communication and storage complexity point of view, the third, Convergent encryption based solution is only slightly worse than the second one because of the symmetric keys that have to be additionally stored. From a computational point of view, the big advantage of the third solution is the fast encryption and decryption of chunks, but a substantial amount of work is transferred to the (mobile) client which might outweigh this advantage.

IV. CONCLUSIONS

In the FIGARO project we developed an architecture for a federation of home gateways. We distinguished inner and outer circle friends and described a mechanism for social-aware backup of content and sharing with friends. To facilitate remote access, content can be segmented and redundantly stored at gateways of friends.

We developed innovative solutions that allow secure and redundant storage of segments with inner and outer circle friends. They enable decoding of segments in the encrypted domain without the need of sharing a decryption key with friends, which relieves the computational effort of mobile devices accessing the content. We combined Vandermonde-Reed-Solomon codes with homomorphic encryption to three different solutions and described their differences in complexity.

ACKNOWLEDGEMENT

This work has been financed by European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2009-5- 258378 (FIGARO project).

REFERENCES

[1] TRIBLER: a social-based peer-to-peer system. Pouwelse, Johan A., et al. Concurrency and Computation: Practice and Experience 20.2 (2008): 127-138.

[2] Distributed Content Backup and Sharing using Social Information. J. Jiang, C. Casetti. IFIP Networking 2012, Prague, Czech Republic, May 2012.

[3] Friendstore: Cooperative online backup using trusted nodes. D. N. Tran, F. Chiang, and J. Li. In SocialNets '08: Proceedings of the 1st workshop on Social network systems, 2008

[4] Holographic dispersal and recovery of information. F. P. Preparata. IEEE Transactions on Information Theory, 35(5):1123–1124, September 1989.

[5] A survey of homomorphic encryption for nonspecialists, Caroline Fontaine, Fabien Galand. EURASIP Journal on Information Security archive, Volume 2007, January 2007, Article No. 15,Hindawi Publishing Corp. New York, NY, United States

[6] A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, James S. Plank, Technical Report CS-96-332, Department of Computer Science, University of Tennessee.

[7] Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, Pascal Paillier, Editor: J. Stern, Advances in Cryptology, EUROCRYPT'99, vol. 1592 of Lecture Notes in Computer Science, pp. 223-238, Springer-Verlag, 1999.

[8] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, R. L. Rivest, A. Shamir, and L. Adleman, Communications of the ACM , vol. 21, No. 2, February 1978.

[9] Reclaiming Space from Duplicate Files in a Serverless Distributed File System, John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, Marvin Theimer, 22nd International Conference on Distributed Computing Systems, IEEE, July 2002.

[10] A Distributed Data Storage and Retrieval Scheme in Unattended WSNs Using Homomorphic Encryption and Secret Sharing, Yi Ren, Vladimir Oleshchuk, and Frank Y. Li, WD'09 Proceedings of the 2nd IFIP conference on Wireless days, pages 282-287, IEEE Press Piscataway, NJ, USA, 2009.

[11] Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance, Qian Wang, Kui Ren, Wenjing Lou and Yanchao Zhang, IEEE INFOCOM 2009.