PE-AONT: Partial Encryption All or Nothing Transform

Katarzyna Kapusta[1,2][a] and Gerard Memmi[2][b]
[1]*Thales SIX GTS France, ThereSIS, Palaiseau, France*
[2]*LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France*

Keywords: Data Fragmentation, Data Dispersal, Distributed Storage, Multi-storage, Cloud Storage Security, Key Exposure, Secret Sharing, All-or-Nothing, AONT, SAKE-secure.

Abstract: We introduce PE-AONT: a novel algorithm for very fast computational secret sharing scheme. The core idea of this scheme is to encrypt the data only partially before applying an all-or-nothing transform that will blend the encrypted and non-encrypted data. By doing this, we achieve much better performance than relevant techniques including straightforward encryption. To this regard, a performance benchmark is provided. Interestingly, when the ratio between the number of encrypted and non-encrypted fragments is wisely chosen, data inside fragments are protected against exposure of the encryption key unless all fragments are gathered by an attacker. Therefore, by choosing the right parameters, we can achieve key exposure protection, faster processing, and a better overall protection.

1 INTRODUCTION

Fragmenting and dispersing data over multiple independent storage sites reinforces its levels of confidentiality, integrity, and availability (Bessani et al., 2013). During the past decades, multiple fragmentation methods were defined and used in various settings (Buchanan et al., 2015; Memmi et al., 2015). These methods include but are not limited to information-theoretic secret sharing schemes (Shamir, 1979; Blakley, 1979), information dispersal algorithms (Rabin, 1989), as well as schemes combining symmetric encryption with data fragmentation (Krawczyk, 1994; Resch and Plank, 2011).

In distributed systems, it is not uncommon to encrypt then fragment data in large chunks in a straightforward manner and then disperse them over different independent servers or storage sites. To support key management, encryption keys will be fragmented using a secret sharing scheme and appended to data fragments. To ensure resilience, data replication or error-correction codes (Reed and Solomon, 1960) may be applied. This sturdy way of proceeding does not protect against attackers able to recover encryption keys. Such attackers may obtain a portion of information just by decrypting fragments they succeeded to access to.

Protection against key exposure requires additional post-processing that will create dependencies between ciphertext blocks contained inside the fragments (Karame et al., 2018) or that will shred ciphertext blocks over fragments (Kapusta and Memmi, 2018a). Encrypted data are then protected against key exposure unless all fragments are gathered. Such additional protection comes at a performance cost, as it increases the required number of operations.

On another hand, partial encryption is usually used in lightweight cipher. In this paper, we introduce Partial Encryption with All Or Nothing Transform (PE-AONT) : a new fragmentation algorithm that is not only faster than most common fragmentation techniques, but also (when carefully applied) protects encrypted data inside fragments against a situation of key exposure. To this end, PE-AONT fragments initial data into k fragments and encrypts only e of them. In a next step, it applies an AONT over the totality of the fragments that exclusive-ors ciphertext with plaintext blocks. PE-AONT is then combining AONT technique with partial encryption achieving both speed and a good level of security. A secure dispersal of fragments ensures that data from a single fragment cannot be recovered unless all fragments are being gathered. Limiting data encryption to e fragments allows speeding up performance in proportion of $(k-e)/k$, while the AONT combined with fragmentation ensures good level of data protection. Last but not least, tuning the values of k and e provides

[a] https://orcid.org/0000-0003-0963-4782
[b] https://orcid.org/0000-0002-3380-8394

Kapusta, K. and Memmi, G.
PE-AONT: Partial Encryption All or Nothing Transform.
DOI: 10.5220/0011300600003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 636-641
ISBN: 978-989-758-590-6; ISSN: 2184-7711

the user with a seldom means to easily change the ratio between performance and data protection level in order to meet the desired requirements of his application.

Outline. In Section 2, we present a selection of relevant techniques for fast data fragmentation. We also recall two threat models to analyze protection under key exposure. In Section 3, we describe in details the proposed PE-AONT algorithm. In Section 4, we compare PE-AONT with relevant works, analyzing its complexity and level of data protection. A performance benchmark is provided in Section 5. We conclude with an insight into future works.

2 RELEVANT WORKS

In this section, we briefly describe selected relevant fragmentation techniques that will later be used for theoretical ans performance comparisons. The interested reader can go to (Memmi et al., 2015; Qiu et al., 2019) for more extensive surveys.

2.1 Notations

By convention, we keep upper cases variables for sequences of bits (D for the initial data, F for a fragment, C for a block). Lower cases variables are for parameters: initial data D are divided into k fragments. A fragment is seen as sequence of f blocks and D of $l = f \times k$ blocks.

For resilience purposes, a threshold (i.e. the minimal number of fragments necessary to reconstruct D, moreover, less fragments must not provide any information about the D) can be introduced, In such a case, D is divided into n fragments with a threshold of k.

2.2 Encryption and Straightforward Fragmentation

A computationally secure scheme for fragmentation of a large amount of data was first introduced by Krawczyk in his seminal work *Secret Sharing Made Short* (SSMS) (Krawczyk, 1994). SSMS is a general methodology combining data encryption and fragmentation together with secret sharing of the encryption key. In the original proposal, data is encrypted using a symmetric cipher (usually, AES), and then fragmented using an information dispersal algorithm (e.g. see (Rabin, 1989)) into n fragments. A threshold of k of them are needed for data reconstruction. Encryption key is split using a perfect secret sharing scheme (typically Shamir's secret sharing (Shamir, 1979)) and embedded within data fragments. In contrast to perfect secret sharing schemes (such as one-time pad or Shamir's scheme), the storage overhead of SSMS does not depend on data size, but is equal to the key size per data fragment which is practically negligible for sizeable data.

Performance of SSMS depends on implementation details of encryption and dispersal techniques. In modern implementations, systematic error-correction codes are used as the dispersal technique (Bessani et al., 2013; Reed and Solomon, 1960). This improves the performance, but allows a partial decryption of compromised fragments in a situation of key exposure, as the set of fragments contains fragments that are formed from large encrypted data chunks.

Alike SSMS, the AONT-RS method (Chen et al., 2017; Resch and Plank, 2011) combines symmetric encryption with data dispersal. The difference between those two methodologies lies in the key management. In AONT-RS, the key is exclusive-ored with the encrypted data hash. See (Qiu et al., 2019) for a survey on different AONT variations such as (Boyko, 1999). A recent extension to AONT can be found in (Esfahani et al., 2021).

2.3 CAKE Threat Model and the Bastion Scheme

The Ciphertext Access under Key Exposure (CAKE) threat model (Karame et al., 2018) aims at addressing the problem of data protection against key exposure in cloud environments. A scheme is denoted to be $(l-\lambda)$ CAKE-secure when it resists an attacker able to access *any* $(l-\lambda)$ blocks of the dispersed ciphertext as well as the encryption key. Bastion protects transformed ciphertext against key exposure as long as two blocks are not being exposed ((l-2) CAKE-secure).

Authors of the CAKE model introduce an efficient encryption scheme, Bastion, that provides $c-2$ CAKE security (decryption computationally infeasible unless all but two ciphertext blocks are being gathered). In more details, the sequence of l input ciphertext blocks $X = X_1, \ldots, X_l$ is is transformed into a sequence of l transformed blocks: $X' = X'_1, \ldots, X'_l$ by multiplying X by a square matrix A, such that: (i) all diagonal elements are set to 0, and (ii) the remaining off-diagonal elements are set to 1. The multiplication $X' = A \cdot X$ (where 'and' and 'xor' stand for 'multiply' and 'add' respectively) ensures that each output block X'_i will depend on all input blocks X_j excluding X_i. k fragments are then formed from the transformed ciphertext, for instance, by cutting it into blocks of $\frac{l}{k}$.

The Bastion scheme achieves much better performance than other AONTs as it requires only $2l$

exclusive-or operations in addition to data encryption.

2.4 SAKE Threat Model

The Shares Access under Key Exposure (SAKE) threat model introduced in (Kapusta et al., 2020), is a variation of CAKE that results from the observation that once an attacker compromises a storage site, she will be most probably able to acquire all the blocks of the share S_i stored at this site. A scheme is denoted $(k-\lambda)$ SAKE-secure when it resists an attacker able to access *any* $(k-\lambda)$ fragments of the dispersed ciphertext as well as the encryption key. The SAKE model comes with SSAKE and ROSSAKE schemes that outperforms the Bastion scheme (thanks to changing the threat model, it is possible to reduce (SSAKE) or even remove (ROSSAKE) the post-processing of the ciphertext) (see (Kapusta et al., 2020) for details).

2.5 Mix and Slice

Mix&Slice (Bacis et al., 2016) is an approach to enforce access revocation on data stored at external Cloud providers. Before being uploaded to the cloud, data is transformed in an AONT manner making the data decryption infeasible as long as the ciphertext is incomplete. Consequently, re-encrypting even a small portion of the outsourced data with a fresh key revokes the access to a user who does not possess the new key. Mix&Slice uses a symmetric block cipher to create dependencies inside the data. Indeed, a symmetric block function transforms a plaintext block into a ciphertext block, correct decryption of which is infeasible when even a single bit of data is missing. Thus, it is possible to use it as a way of creating strong dependencies between bits of data.

2.6 Secure Fragmentation and Dispersal (SFD)

Secure fragmentation and dispersal (Kapusta and Memmi, 2018a) divides a ciphertext encrypted using a block cipher with a mode of operation that creates chaining between consecutive ciphertext blocks (like Cipher Block Chaining) into fragments resisting to key exposure. The complete scheme is composed of three steps. The first step separates consecutive blocks of the ciphertext. The second step separates bits of blocks over final fragments. At last, fragments are dispersed and stored over independent storage locations e.g. multiple cloud providers (like in (Bessani et al., 2013)). In consequence, an attacker present at a single storage location is unable to decrypt a single block of the ciphertext even if she possesses the right encryption key.

3 ALGORITHM DESCRIPTION

PE-AONT is composed of three steps detailed in the subsequent subsections. In a first step, initial data D together with an Initial Vector (IV) is composed of l plaintext blocks. They are structured into k fragments of which only e of them are encrypted. In a second step, the k fragments are transformed using an AONT that blends ciphertext contained inside the e encrypted fragments with the $k-e$ plaintext fragments in a one-time pad fashion. It is this step that is ensuring the protection of the $k-e$ plaintext fragments with a level of protection depending on the values of k and e. The last step simply consists of dispersing the fragments.

3.1 Step1: Data Fragmentation and Partial Encryption

The pseudo-code of the first step of the algorithm - FRAGMENTANDENCRYPT- is presented in Figure 1 and illustrated by the example in Figure 2.

1: **function** FRAGMENTANDENCRYPT(D, e, k)
2: Fragment data D into k initial fragments $F_0, \ldots, F_{k-1}$ and of
3: //each fragment has f blocks
4: //Encrypt e of the k fragments:
5: **for** each fragment F_i, $i = 0, \ldots, e-1$ **do**
6: Encrypt fragment F_i
7: // l blocks $C_0, \ldots, C_{l-1}$ are now straightforwardly defined from the e encrypted fragments and the $k-e$ plaintext fragments.

Figure 1: Pseudo-code of Step 1 fragmenting data into k fragments, encrypting e of them, creating l blocks.

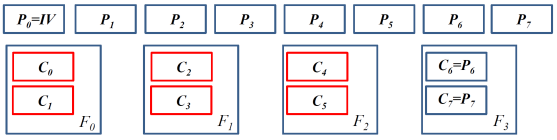

Figure 2: Step 1 example with $k = 4$, $e = 3$, and $l = 8$. The first step of the algorithm fragments D into k fragments of $f = 2$ blocks, the last fragment contains plaintext blocks. D has 7 blocks: $D = P_1, P_2, ...P_7$. The block $P_0 = IV$ is added to D; it corresponds to the ciphertext block containing the Initial Vector used by the cipher.

During this step, a block P_0 containing the Initial Vector (IV) is added to $D = P_1, P_2, ...P_l$ the initial data to form l plaintext blocks which are structured into k fragments. Only $e < k$ of the fragments are being en-

crypted using a block cipher (usually, AES). Therefore, at the end of the processing l blocks denoted with the letter C are created. Each fragment contains $f = \frac{l}{k}$ blocks, e fragments contain ciphertext blocks, $k-e$ of them contain plaintext blocks.

The fragments can be put together side by side to form a matrix of f rows and k columns as in Figure 2 where each block is an element of the matrix, each column a fragment. This way of organizing blocks will be used to more easily describe the second step of our algorithm.

3.2 Step 2: All-or-Nothing Transform of the Fragments

The pseudo-code of the second step of the algorithm - AONTTRANSFORM - is presented in Figure 3 and illustrated by a continuation of the example of Figure 2 shown in Figure 4. Blocks are processed in a *row by row* fashion by sets of k blocks, each block coming from a different fragment. The AONT presented in (Karame et al., 2018; Stinson, 2001) is applied over the set of k blocks, e of which are ciphertext blocks and the $k-e$ remaining ones are plaintext blocks.

1: **function** AONTRANSFORM(D)
2: **for** $i = 0, \ldots, f-1$ **do**
3: $sum = \bigoplus_0^{j=k-1} C_{f\times j+i}$
4: **for** $j = 0, \ldots, k-1$ **do**
5: $C_{f\times j+i} = sum \oplus C_{f\times j+i}$

Figure 3: Pseudo-code of Step 2. An AONT is applied over the k fragments exclusive-oring their plaintext and ciphertext blocks. f denotes the number of blocks inside a fragment. Blocks are processed by 'rows' of k blocks (each block of a given row coming from a different fragment). For each 'row' of k blocks a *sum* value containing the exclusive-or of the k blocks is first computed. *sum* is then exclusive-ored with each original block in the *row* in order to remove it and getting a different new block in each fragment.

It must be pointed out that for $k = 2$ this processing does not change anything about the values of the blocks save for the 2 fragments numbering. For $k = 3$ the knowledge of 2 fragments allows deducing the third one. It is therefore, necessary to consider $k > 3$.

3.3 Step 3: Dispersing Fragments

In a final step, fragments are dispersed over independent storage sites. The dispersal technique depends on values of e and k and of course, on the number of available machines.

- For $e = k-1$: no more than $k-2$ fragments can be stored at a single storage site. This comes from the property of the linear AONT protecting transformed blocks unless more than $k-2$ blocks are exposed (Karame et al., 2018).
- For $e < k-1$: all fragments have to be dispersed over independent storage sites. As the number of plaintext fragments increases, it is possible that same combinations of ciphertext blocks will be used to protect different plaintext blocks.

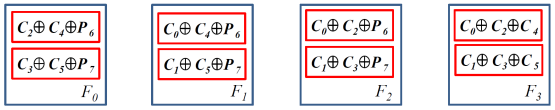

Figure 4: Step2, example for $k = 4$ and $e = 3$. Fragments are transformed using an AONT that exclusive-ors ciphertext and plaintext blocks. The transform is applied 'row' by 'row' over k blocks (each block comes from a different fragment) protecting the $k-e$ fragments which were not encrypted.

3.4 Further Considerations

As -AONTransform- uses the Bastion transform over k blocks at a time, k has to be even in order to allow the Bastion's reconstruction matrix to be invertible (Karame et al., 2018)). Under this condition, the defragmentation process is an inverse of the fragmentation and is straightforward.

Indeed, data protection level is lower as e decreases and fragments are only secure when the assumed attacker is not able to compromise more than one storage site. It is the price that has to be paid for speeding up the algorithm processing.

4 THEORETICAL ASPECTS

4.1 Choosing PE-AONT Parameters

To achieve the protection against key exposure, each plaintext block should be exclusive-ored with at least two different ciphertext blocks. This goes from the definition of a SAKE-secure scheme, which is presented in details in (Kapusta et al., 2020). An intuitive explanation is that an adversary possessing a linear combination of plaintext blocks xored with a single ciphertext block is able to win the ind-CPA game. In contrary, if the plaintext blocks are xored with more that one ciphertext blocks, the adversary has a negligible chance to win the game, even if it possess the right encryption key.

Thus, the minimum value of e is 3, which enables combinations of at least two ciphertext blocks hiding a plaintext. The minimum value of k is 4, as the number of input block has to be even in order to allow

the Bastion's reconstruction matrix to be invertible (Karame et al., 2018) and $k \geq e$. Intuitively, the larger the value of e, the higher will be the provided level of data protection. When $e = k - 1$, data cannot be decrypted unless all k fragments are gathered. Fragments are thus protected against key exposure (are SAKE-secured). It's the most interesting PE-AONT configuration from a security point of view, as it is not only faster than encryption and straightforward fragmentation (enc.+ SF) but also protects data against key exposure.

When $3 \leq e \leq k-2$ the protection is lower and ensured only if the attacker is not able to access to more than a single storage site. Indeed, when the attacker compromised more than one site, she can obtain relationships between fragments by exclusive-oring the fragments in her possession (a similar problem occurs when a one-time pad is reused).

4.2 Comparison with Relevant Schemes

We compared PE-AONT with relevant works in terms of amount of computations and ability to protect against key exposure. Results are shown in Figure 5. As a baseline, we use encryption that requires $l-1$ block cipher operations and $l-1$ exclusive-or operations (when applied on a plaintext composed of $m = l-1$ blocks).

Algorithm	Block op.	Exclusive-ors	K.E.P.
Encryption	l-1 b.c.	l-1	No
Bastion	l-1 b.c.	3l-1 [AON Transform: 2l]	Yes
SFD	l-1 b.c.	l-1	Yes
PE-AONT	$e \times f$ b.c.	$2l + f(e-1)$	**Yes***

Figure 5: Comparison in terms of number of block cipher operations (block op.), number of exclusive-ors, and ability to provide a key exposure protection (K.E.P), assuming a plaintext of $l-1$ blocks. (*): PE-AONT provides key exposure protection when $e \geq 3$. When $e < k-1$ fragments are protected against key exposure, but data is protected only against an attacker at a single storage site.

Bastion scheme applies only a linear transform over the encrypted data without increasing the number of block cipher operations. Bastion's transform uses $2l$ exclusive-or operations. Counting with the encryption step, Bastion scheme requires $3l-1$ exclusive-ors.

SFD does not require additional operations in addition to data encryption, as it just disperses data chunks over fragments (depending on the programming technique this also leads to a slight performance overhead of up to 11%).

PE-AONT requires only $e \times f$ block cipher operations, as it does not encrypt the totality of the data contained inside the fragments. It performs $2l + f \times (e-1)$ exclusive-or operations: $e \times f$ during the encryption and $2l - f$ during the AONT.

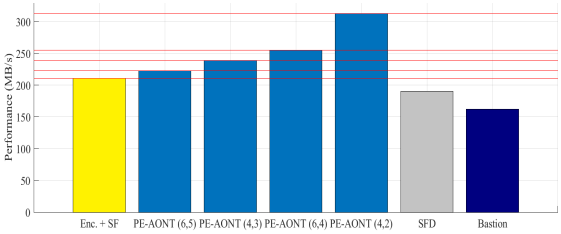

Figure 6: Performance benchmark. PE-AONT was measured in different configurations of (k,e). In all of them it is faster than encryption and straightforward fragmentation (enc.+ SF). It is also faster than two fragmentation schemes protecting against data exposure: secure fragmentation and dispersal (SFD) and the Bastion scheme.

5 PERFORMANCE COMPARISON

Implementation Details. Relevant algorithms were implemented using the same programing style in JAVA with JDK 1.8 on DELL Latitude E6540, X64-based PC running on Intel® Core™ i7-4800MQ CPU @ 2.70 GHz with 8 GB RAM, under Windows 7. Standard *javax.crypto* library was used. A random data sample was used for each measurement and each presented result is an average of 30 measurements. AES-NI with 128 bits key was used for encryption.

Performance comparison between relevant algorithms is presented in Figure 6. The performance of PE-AONT is shown in 4 configurations: two configurations were $e = k-1$ (and where the protection level is high) and for two configurations when $e \leq k-2$ (data protection is traded for better performance). In all configurations, PE-AONT outperforms encryption and straightforward fragmentation. Moreover, it is much faster than two relevant fragmentation techniques protecting fragmented data against key exposure: SFD and Bastion's scheme.

6 FUTURE WORKS

Our software integrates AES-NI cipher and PE-AONT in a coarse-grained fashion. In the future, we would like to integrate the algorithm within the code of the encryption process in a fine-grain fashion to allow an even better fragmentation performance. We improved the Bastion AONT algorithm by decreasing the number of Xor operations needed to protect

data against key exposure in (Kapusta and Memmi, 2018b) or in (Kapusta and Memmi, 2018c). It becomes important to keep analyzing and comparing these various scheme and determine in which situation one method is preferable over another one. We believe PE-AONT is very likely best fitted as a tunable lightweight cipher since it is faster than the other AONT algorithms.

Moreover, we would like to integrate PE-AONT inside a complete distributed system protecting data using an adaptable combination of fragmentation, encryption, and dispersal (Kapusta et al., 2020).

7 CONCLUSION

PE-AONT: a fast fragmentation method combining Partial Encryption with an AONT was introduced in order to speed up the overall scheme performance in a tun-able fashion. The desired ratio between performance and protection levels can easily be adjusted by users with just two parameters to increase performance and decrease protection or the other way around. An experimental evaluation shows that the algorithm achieves better performance than relevant techniques including the most common way of data fragmentation. As such, it can be chosen as a lightweight cipher. Moreover, by carefully choosing parameters values, it protects fragmented data against key exposure (SAKE-secure) as long as an attacker does not access to the entire set of fragments.

We now believe that PE-AONT is ready for serious consideration as a component and successfully be integrated within modern distributed or transmission systems where rapid protection is required.

REFERENCES

Bacis, E., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Rosa, M., and Samarati, P. (2016). Mix&slice: Efficient access revocation in the cloud. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*, CCS'16, pages 217—-228, New York, NY, USA. ACM.

Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2013). Depsky: Dependable and secure storage in a cloud-of-clouds. *Trans. Storage*, 9(4):12:1–12:33.

Blakley, G. (1979). Safeguarding cryptographic keys. In *Proc. of the National Computer Conf. v48, pp313-317*.

Boyko, V. (1999). On the security properties of OAEP as an all-or-nothing transform. In Wiener, M. J., editor, *Advances in Cryptology - CRYPTO'99, 19th Annual Int. Cryptology Conf., Santa Barbara, CA, USA, August, 1999, Proc.*, LNCS 1666, P503-518. Springer.

Buchanan, W., Lanc, D., Ukwandu, E., Fan, L., and Russell, G. (2015). The future internet: A world of secret shares. *Future Internet*, 7(4):445.

Chen, L., Laing, T. M., and Martin, K. M. (2017). *Revisiting and Extending the AONT-RS Scheme: A Robust Computationally Secure Secret Sharing Scheme*, pages 40–57. Springer Int. Pub., Cham.

Esfahani, N. N., Cheriton, D. R., and Stinson, D. R. (2021). Asymetric all-or-nothing transforms. *arXiv:2105.14988v1*.

Kapusta, K. and Memmi, G. (2018a). Enhancing data protection in a distributed storage environment using structure-wise fragmentation and dispersal of encrypted data. In *2018 17th IEEE Int. Conf. On Trust, Security And Privacy In Computing And Communications (TrustCom/BigDataSE)*, pages 385–390.

Kapusta, K. and Memmi, G. (2018b). Poster: Circular aon: A very fast scheme to protect encrypted data against key exposure. In *ACM CCS'18*, Toronto, Canada.

Kapusta, K. and Memmi, G. (2018c). Selective all-or-nothing transform: Protecting outsourced data against key exposure. In *Proc. of 10th Int. Symp, CSS, LNCS 11161*, pages 181–193, Amalfi, Italy. Springer.

Kapusta, K., Rambaud, M., and Memmi, G. (2020). Revisiting shared data protection against key exposure. In Sun, H., Shieh, S., Gu, G., and Ateniese, G., editors, *ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020*, pages 165–177. ACM.

Karame, G. O., Soriente, C., Lichota, K., and Capkun, S. (2018). Securing cloud data under key exposure. *IEEE Trans. on Cloud Computing*, pages 1–1.

Krawczyk, H. (1994). Secret sharing made short. In *Proc. of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 136–146, London, UK. Springer-Verlag.

Memmi, G., Kapusta, K., and Qiu, H. (2015). Data protection: Combining fragmentation, encryption, and dispersion. In *2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)*, pages 1–9.

Qiu, H., Kapusta, K., Lu, Z., Qiu, M., and Memmi, G. (2019). All-or-nothing data protection for ubiquitous communication: Challenges and perspectives. *Inf. Sci.*, 502:434–445.

Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348.

Reed, I. S. and Solomon, G. (1960). Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304.

Resch, J. K. and Plank, J. S. (2011). Aont-rs: Blending security and performance in dispersed storage systems. In 9th *USENIX Conf. on File and Storage Technologies*, FAST'11, pages 14–14, Berkeley, CA, USA.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22(11):612–613.

Stinson, D. R. (2001). Something about all or nothing (transforms). *Des. Codes Crypto.*, 22(2):133–138.