This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

Investigating the Applicability of Nested Secret Share for Drone Fleet Photo Storage

Antonino Galletta*†‡§¶, *Member, IEEE*, Javid Taheri†, *Senior Member, IEEE*, Antonio Celesti†, Maria Fazio†, and Massimo Villari*†, *Senior Member, IEEE*
* MIFT Department, University of Messina, Italy
{angalletta, acelesti, mfazio, mvillari}@unime.it
† Computer Science Department, University of Karlstad, Sweden
{antonino.galletta, javid.taheri}@kau.se
‡ On Behalf of INDAM, GNCS (Gruppo Nazionale per il Calcolo Scientifico)
§ On Behalf of CINI, Laboratorio Informatica e Società (IeS)
¶ Corresponding author: Antonino Galletta email: angalletta@unime.it

Abstract—Military drones can be used for surveillance or spying on enemies. They, however, can be either destroyed or captured, therefore photos contained inside them can be lost or revealed to the attacker. A possible solution to solve such a problem is to adopt Secret Share (SS) techniques to split photos into several sections/chunks and distribute them among a fleet of drones. The advantages of using such a technique are two folds. Firstly, no single drone contains any photo in its entirety; thus even when a drone is captured, the attacker cannot discover any photos. Secondly, the storage requirements of drones can be simplified, and thus cheaper drones can be produced for such missions. In this scenario, a fleet of drones consists of t+r drones, where t (threshold) is the minimum number of drones required to reconstruct the photos, and r (redundancy) is the maximum number of lost drones the system can tolerate. The optimal configuration of t+r is a formidable task. This configuration is typically rigid and hard to modify in order to fit the requirements of specific missions. In this work, we addressed such an issue and proposed the adoption of a flexible Nested Secret Share (NSS) technique. In our experiments, we compared two of the major SS algorithms (Shamir's schema and the Redundant Residue Number System (RRNS)) with their Two-Level NSS (2NSS) variants to store/retrieve photos. Results showed that Redundant Residue Number System (RRNS) is more suitable for a drone fleet scenario.

Index Terms—Drones, Secret Share Algorithms, Nested Secret Share Algorithms, Redundant Residue Number System, Shamir schema.

1 Introduction

Drones are becoming very popular nowadays and are often used for military and civil purposes including teaching [1], photography, surveillance, rescue operations, spying on enemies, monitoring of infrastructure, farming, and aerial mapping [2].

Drones are also often destroyed [3] or captured by enemies [4], and so photos stored inside could be lost or revealed.

To solve such a problem, a solution is to adopt Secret Share (SS) algorithms to split taken photos into several sections/chunks and distribute them among a fleet of drones. Well-known SS algorithms that can do this task include Blackey, Shamir, and Redundant Residue Number System (RRNS). Based on these algorithms, several techniques, such as proactive ones, were developed to improve the security of the whole system.

A typical solution is to compose a drone fleet with $t+r$ drones, where t (threshold) is the minimum number of drones required to reconstruct all photos, and r (redundancy) is the maximum number of lost drones that the system can tolerate. The $t+r$ configuration is typically a formidable task because it is not possible to fine-tune related algorithms to perfectly fit all requirements of any specific mission, ranging from "Unclassified" (no secret) to "Top Secret" [5]. In fact, the value of the threshold t depends greatly on the adopted algorithm and cannot be changed (in almost all cases) according to specific application scenarios.

The objective of this paper, hence our contribution to the field, is to address this problem and propose an innovative/flexible SS-based technique based on the Nested Secret Share (NSS) schema [6]. In this paper, we assess the applicability of such a technique for drone fleets. In particular, we consider the 'security of NSS', the 'mission time', and the 'energy' requirements. NSS cascades different SS levels where the output of an SS level becomes the input to another level. Using such a technique, it is possible to build n-SS levels. One of the strengths of our NSS technique is its generality; that is, it can be applied to any SS algorithm that fits the requirements of any specific application scenario. Therefore, the main contributions of this scientific work are as follows.

- Proposing the NSS strategy, elaborating on how it performs in different circumstances, and analyzing its performance to identify its pros and cons.
- Finding the best configuration to use NSS algorithms for storing photos in drone fleets.
- Compare the performances of two major SS algorithms (Shamir's schema and the Redundant Residue Number System) and measure their suitability for building Two-Level NSS (2NSS) variants.

In our experiments, we considered different types of missions (ranging from "Unclassified" to "Top Secret"), and various photo sizes, and analyzed each NSS variant for its provided security level, execution time, and power consumption. The results confirmed that 2NSS is much more flexible than SS, where lower redundancy values r lead to more secure storage systems (w.r.t.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

a single-layer SS solution); in other words, an adversary/enemy must steal more drones to construct the recorded photos (or any other sort of information).

The remainder of the paper is organized as follows. Section 2 describes related works and highlights the added value of our study. The threat model is discussed in Section 3. Section 4 presents basic SS techniques. Section 5 discusses the mathematical foundation of NSS. Section 6 describes the implementation of the developed system. Section 7 compares and assesses NSS. Section 8 concludes the paper.

2 Related Work

The objective of this article is to apply an SS-based technique to build a secure and reliable drone fleet photo (or any other sort of information) storage management. Because no similar related works exist in the literature, at the time of writing this paper, we focus on analyzing the major SS and drone photo storage management solutions.

2.1 Secret Share (SS)

Secret Share (SS) schemes have already been used in various scientific communities to secure different application scenarios. A Reversible Data Hiding Scheme (RDHS) for images is based on the Chinese Remainder Theorem [7]. The proposed system allows sharing an image between a sender and a receiver while embedding additional data by using two techniques based on homomorphic encryption and difference expansion. The first method allows the data extraction after the recomposition of the image; the latter supports the data extraction from shares without the need to reconstruct the image.

A storage system based on SS has been discussed in [8]. The authors proposed an RRNS-based system to distribute shares among several public cloud storage services (like Google Drive, Dropbox etc). They could recompose the original final even when some cloud storage services become unavailable.

An assessment of different SS approaches in Cloud, Edge and IoT domains has been presented in [9]. The authors tested RRNS and Shamir's schema to verify the applicability of such techniques. Through comprehensive experiments, they discovered that the computation (split and merge) of files bigger than $50MB$ should be carried out on Cloud resources, from $500kB$ to $50MB$ in Edge devices, and the rest (ie, smaller files) on IoT devices.

A secure key exchange system for IoT devices has been discussed in [10]. The authors proposed an SS-based software to make a secure communication through MQTT (Message Queue Telemetry Transport) without using the SSL/TLS feature. The basic idea was to share a secret key (split by means of SS) before starting the communication of data.

A Visual Secret Sharing schema with noise correction has been presented in [11]. The proposed approach claims to remove the noise only if 1 of the k shares is affected. The proposed approach cannot identify the corrupted share. The noise correction process is executed on the recomposed image, and per pixel, calculates the average value of a $3x3$ matrix containing its neighbors.

A Verifiable Secret Sharing schema with cheater identification has been discussed in [12]. The advantage of the proposed systems, compared to others like [13], [14], is the ability to discover the corrupted shares before secret recomposition, thus reducing the computation time required to rebuild the secret.

A Secret Sharing-based Threshold Password Authentication system has been proposed in [15]. The system is made up of two servers to store shares and a gateway to recompose the password. To retrieve the secret, users send an encrypted request to the gateway, which in turn forwards it to the servers. The two servers will generate the verification data and send it to the Gateway for the reconstruction of the secret.

A Block-based Progressive Visual Secret Sharing (BPVSS) solution to reduce the risk of intruders during the transmission of images has been proposed in [16]. The authors combined the well-known BPVSS with the Least Significant Bits steganography technique. The basic idea is to hide the shares generated from BPVSS within cover images in such a way that it would not be possible (for human eyes) to find artifacts in the shares.

A Shamir scheme's-based framework for the aggregation of queries in distributed Database-as-a-Service has been proposed in [17]. The proposed framework differs from common Database Management Systems (DBMS) as it does not allow executing indexing and optimizing procedures. In this approach, a range-based query is split into equality queries. Despite these limitations, the proposed system is robust against several attacks, including some of the most challenging ones such as the *honest but curious* user.

2.2 Photo Storage Platforms for Drones

A photo encoding and communication technique based on the Reed Solomon algorithm has been proposed for drones in [18]. The authors discussed a 4-bit encoder to correct 2 errors per block. To verify the correctness of the decoded picture, they created a new image starting from the checksum values. If the new image is full black, then there are no errors. In [19], a novel drone-based video surveillance system is proposed. By using drone features (such as autonomous flight capabilities) and processing images (by using Probability Hypothesis Density algorithms), the authors discussed how to send fleets of drones to track human interactions. In [20] a surface water monitoring system is proposed using drones. The authors modified the drones to implement several functions, including collecting water samples, sending real-time video feeds to the base station; and studying the effect of wind & water parameters. Another drone-based video surveillance system has been proposed in [21]. The authors incorporate motion features into commercial drones. A compression algorithm that, by using the predicted drones' trajectory, sends only "important frames" to the base station is proposed in [22]. The goal of this algorithm is to minimize the usage of bandwidth, allowing the real-time processing of images on the base station. A system for confidentially sending video from drones to the cloud has been proposed in [23]. The authors used homomorphic encryption techniques to encode (cipher) the videos before sending them to the cloud. The algorithm allows reducing the data transmission by sending only the changing foreground parts.A data acquisition and logger system that allows drones to store data during flight has been proposed in [24]. The system is used for weather forecasting. The authors created a waterproof box containing a Raspberry Pi with an SD card to collect humidity, wind speed, and temperature data from sensors.

Although all aforementioned works treated different aspects of drone photo storage management, they do not address any issues regarding captured or destroyed drones. That is, no solutions are provided for cases when a drone is captured by an adversary

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

(enemy) where the captured images must not be revealed, or when a drone is destroyed where the collected photos are simply lost.

3 Threat Model

Let us consider an attacker that aims to spy on an adversary (enemy), for example, Company A spying on Company B. To conduct the attack, the attacker could use drones to take pictures and communicate with each other. The attacker has two main goals: (a) the attacker wants to spy on the adversary without disclosing any information about the spied place (because the adversary could increase the security of the same), and (b) the attacker wants to take and retrieve the maximum number of photos because some drones could be captured and/or destroyed. The goals of the company under attack can be summarized as follows: (a) capture as many drones as possible to discover what the attacker is looking for, (b) destroy drones that cannot be captured, and (c) create an adversarial environment to block/jam communication among drones.

To avoid disclosing any information about the spied asset, the attacker could use an approach similar to ransomware [25], [26] to encrypt all pictures with a public key and keep the private key in the drone base station. Although effective to some extent, this approach is not suitable, considering the purposes of the attacker, for two main reasons: (a) if the attacker does not encrypt the data by using post-quantum cryptography techniques, the enemy could use quantum computing techniques, similar to the ones discussed in [27], [28], to break the encryption schema. Regardless of the amount of time that a quantum computing technique may take to break the encryption, the simple fact that all data is stored in one device increases the chance of deciphering. Our approach replaces the ”possibility to decipher” to ”impossible to decipher” because no device has all the information, and (2) if any drone is captured or destroyed, the attacker loses all photos contained in it. To avoid the latter issue, the attacker could use public-private keys to encrypt the picture and then distribute the encrypted photo to different drones as backups. However, as we have demonstrated in [9], encrypting the file and creating replicas of the same file, considering the same degree of redundancy, requires more storage/memory when compared with RRNS.

A possible solution is to use SS algorithms for drone fleet storage management. This approach brings multiple advantages, including (a) no single drone stores an entire photo, (b) even if one of the drones is captured the enemy is not able to discover photos, (c) a drone fleet can be composed of $p+r$ drones, where t (threshold) is the minimum number of drones required to reconstruct the photos, and r (redundancy) is the maximum number of lost drones that the system can tolerate. As we have seen in Section 2, this paper is not the first attempt to store data by using SS techniques, however, we have a unique approach to store, protect and retrieve photos from drones' storage. In our approach, an enemy able to capture drones can try to combine chunks together in order to rebuild the pictures. To conduct this type of attack s/he does not need any special equipment, only computational capabilities. To make recomposition of pictures more difficult, each share could be encrypted by using a private-public key algorithm. However, this solution is out of the scope of this paper and will be analyzed in future works.

Figure 1 shows a reference scenario including a fleet of $t+r=8$ drones, where $t=5$ and $r=3$. Each drone holds its own internal storage. For example, When drone 4 takes a photo of an asset (e.g., an industrial plant or a military site), the photo is not stored in one place/storage, but is split into $t+r$ obfuscated chunks, each stored on a different drone. In this figure, drones can communicate with each other using a secure wireless technology (e.g., a secured WiFi network). Our assumption is based on previous studies that showed drones can send files to each other by creating point-to-point wireless communications even in adversarial environments (e.g., due to jamming) [29], [30], [31].

According to a specific SS algorithm, the captured photo can be rebuilt using at least $t=5$ chunks. In our example, even if the enemy is managed to capture up to $r=3$ drones (i.e., drones 6, 7, and 8), it would not be able to discover any photo (or any parts of any photo) even by using quantum computing [32]. In this case, $r=3$ is the maximum number of unavailable drones, according to the configuration, the platform can afford to lose. In other words, having $t=5$ drones returned to the base is enough to rebuild all captured photos by all drones (even the ones that are captured or destroyed).

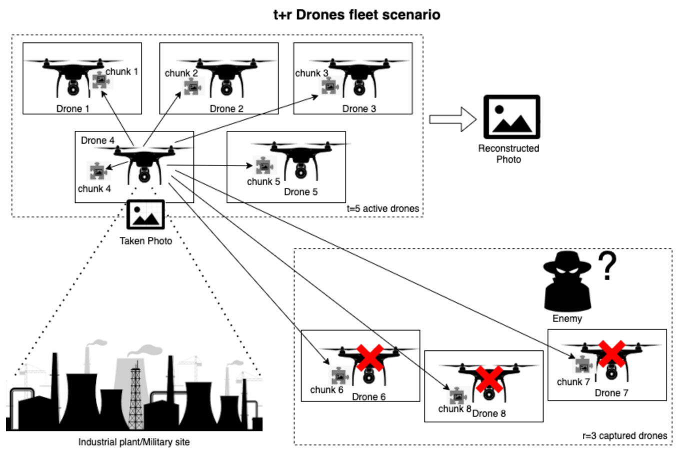


Fig. 1. Fleet of Drones taking and storing photos according to the SS principles.

4 Secret Share Algorithms

Secret Sharing is a well-known technique of cryptography. It has been widely used to securely store data in several domains, including healthcare [33], [34] and Smart Cities [35], [36], [37].

It is a method to encrypt a secret S by splitting it into $n>1$ shares (also called chunks or fragments). Each share $(S_1, S_2, \ldots, S_n)$ contains only some parts of the secret. This means that each of $\binom{n}{n-1}$ combinations of $n-1$ shares does not allow rebuilding S, due to missing data. Under particular conditions, it is possible to recompose the secret by using a subset of t shares, where $t \leq n$. In this case, there would be $\binom{n}{t}$ different possible combinations of shares, all capable of reconstructing the secret S. SS techniques with this feature are also called SS with threshold approach.

The main advantage of a SS algorithm with threshold is the possibility of recomposing the secret even when a certain number of shares become unavailable or corrupted. The maximum number of lost shares can be calculated by subtracting t from n ($r=n-t$), and it is called the “redundancy” degree r ($r \geq 0$)). As it can be inferred, greater values of t represent more confidential algorithms, because the number of shares that are required by an adversary to recompose the secret is larger. We remark that the redundancy

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

degree r does not affect the value of t. SS techniques are different from data partition approaches where the secret S is first partitioned into fragments, and then each partition (chunk) is ciphered. In this case, each chunk contains only parts of the original data, and therefore if an attacker finds the passphrase s/he can retrieve parts of the original secret as well. As discussed earlier, the only way to steal information when using SS is to combine data stored in at least t chunks. This is because each of them does not contain any meaningful information alone. Considering the use case of spy drones (where drones can be captured), the use of SS techniques could reduce the possibility of revealing any recorded information. In the following, we discuss two of the most popular SS algorithms: Shamir's schema and RRNS.

4.1 Shamir's Schema

Shamir's schema [38] was one of the first implementations of SS algorithms. It is based on the mathematical rule that allows retrieving the equation of a polynomial of $k-1$ degree by interpolating the values of k distinct 2D-points (x,y). For example, to retrieve the equation of the following line

$$f(x) = \alpha_0 + \alpha_1 x \tag{1}$$

we only need two points, and to retrieve the equation of the following parabola

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \tag{2}$$

we only need three points; etc.

In general, if we want to split the secret S into t points, we have to consider the equation of a polynomial of $k = t-1$ degrees, that is,

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_{\mathrm{k}} x^{\mathrm{k}} \tag{3}$$

With set $\alpha_0 = S$, we can calculate $f(x)$ using any t distinct points $(i, f(i))$ where $i \in \{1,2,3,\ldots t\}$. In this case, each point $(i, f(i))$ represents a share. If we calculate $f(x)$ using $n = t+r$ distinct points $(i, f(i))$, then we can obtain r degrees of redundancy. The original secret S can be rebuilt by interpolating a subset of t distinct points through the Lagrange interpolation polynomial technique and calculating the value of the equation for $x = 0$ $(f(0))$. The formal validation of Shamir's schema has been provided in [38].

To explain how the algorithm works, we use a simple example (Figure 2) to store/retrieve the secret S using $t = 2$ and $r = 1$. According to this figure, our secret S belongs to the red line. To share it with three parties, we have to calculate and share three points (A, B, and C) that belong to the same line. By interpolating any two points (A and B, B and C, or A and C) through the Lagrange theorem, we can obtain the equation of the red line, and therefore we can retrieve S.

4.2 Redundant Residue Number System

The Redundant Residue Number System (RRNS) [39] is based on the Residue Number System (RNS). RNS chooses t prime moduli called primaries $(m_1, \ldots, m_t)$ such that $m_i > m_{i-1}$ $\forall i \in [1,t]$. RNS is able to represent all the numbers included in the range $[0-M]$, where M can be calculated through the following equation:

$$M = \prod_{i=1}^{t} m_{\mathrm{i}} \tag{4}$$

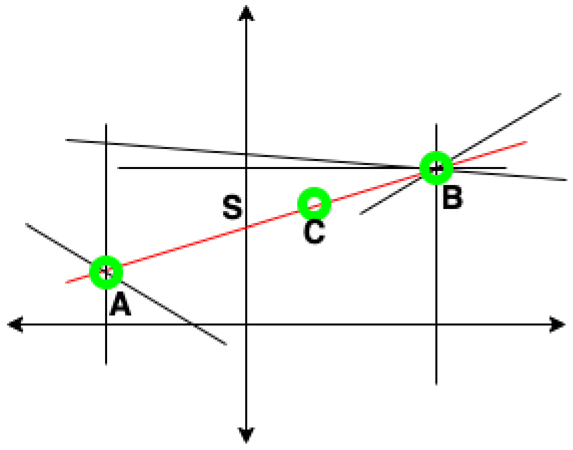


Fig. 2. Example of Shamir's schema.

Considering that the secret S belongs to the range $[0,M]$. The dealer, who wants to calculate the shares and distribute them among peers, has to assign one of the primaries $(m_1, \ldots, m_t)$ to each peer.

$$s_i = S \mod m_i \forall i \in [1,t] \tag{5}$$

To rebuild the secret, the dealer has to combine shares using the Chinese Remainder Theorem (CRT); that is:

$$S = \left(\sum_{i=1}^{t} s_i \frac{M}{m_i} b_i \right) \mod M \tag{6}$$

where b_i, $i \in [1,t]$ is such a way that $\left(b_i \frac{M}{m_i} \right) \mod m_i = 1$.

The RRNS is defined as the RNS. The only difference between RRNS and RNS is the threshold approach. The RNS splits the secret into t shares and needs all of them to rebuild the secret; whereas RRNS is able to divide the secret into $n = t+r$ shares and needs only t of them to reconstruct the secret. This means that any $\binom{n}{t}$ different combinations of shares can be used to reconstruct the secret. The formal validation of RRNS schema has been provided in [39].

In this paper we explain how the algorithm works with an example considering a secret S and a configuration with $t = 2$ and $r = 1$. Let us consider the value of the secret $S = 10$ and three modules (one per each peer) $m_1 = 3; m_2 = 5; m_3 = 7$. Where m_1 and m_2 represent the primaries and m_3 is the redundancy. As we discussed above, to calculate the shares we have to calculate the remainder of the division of the secret S for each modulus. In this case, we will obtain:
$s_1 = 10 \; mod \; m_1 = 10 \; mod \; 3 = 1$
$s_2 = 10 \; mod \; m_2 = 10 \; mod \; 5 = 0$
$s_3 = 10 \; mod \; m_3 = 10 \; mod \; 7 = 3$

To rebuild the secret S we can use the CRT on (s_1, s_2, s_3) by considering the modules (m_1, m_2, m_3). In particular, starting from the value of the shares $(s_1 = 1; s_2 = 0; s_3 = 3)$, we can calculate the Secret by adding to each share the relative modulus several times.
$S_1 = s_1 + x_1 \times m_1$
$S_2 = s_2 + x_2 \times m_2$
$S_3 = s_3 + x_3 \times m_3$
CRT ensure that a triple (x_1, x_2, x_3) that allows us to have

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

$S_1 = S_2 = S_3$ exists and $S_1 = S_2 = S_3 = S$ is the Secret. In our example the triple is $(3,2,1)$.
$S_1 = 1+3\times 3 = 10$
$S_2 = 0+2\times 5 = 10$
$S_3 = 3+1\times 7 = 10$

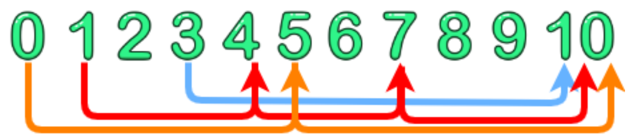


Fig. 3. Example of RRNS merge by using the Chinese Reminder Theorem.

In Figure 3 a graphical example is shown. In the figure, we assigned a different color to each peer (and its relative modulus). Peer 1 ($m_1 = 3$) has the red color; peer 2 ($m_2 = 5$) has the orange color; and peer 3 ($m_3 = 7$) has the blue color. The first common number (the first point where at least two lines intersect in the figure) is the secret. As the reader can observe in the figure, the first common number is 10 that was the value of our secret S.

The main drawback of using RRNS, as compared with Shamir's schema, is related to how the minimum number of shares is set. In RRNS, the minimum number of shares is fixed and depends on the value of primaries; whereas in Shamir's schema, it can be chosen arbitrarily.

5 Nested Secret Share

As we discussed in Section 4, t is the minimum number of chunks needed to reconstruct the original secret S; r also represents the security degree of an SS algorithm. The maximum value of r depends greatly on the specific adopted algorithm, and thus it is not possible to flexibly change it based on the needs of any specific application scenario. Considering our drone fleet photo storage case study, this behavior imposes a limit because it is not possible to choose the best $t+r$ configuration (according to a particular type of mission that can range from "top secret" to "unclassified") at all time. A possible solution to address such a problem is to recursively nest two or more SS algorithms using our Nested Secret Share (NSS) technique. The basic idea behind our NSS is that it considers different SS levels ($n \in 1,2,3,\ldots$), where the output of an upper SS level becomes the input of its lower SS level. One of the strengths of such an approach is its generality; that is, NSS can be applied to any SS algorithm to fitting the requirement of any specific application scenario.

Let the secret data be a value S. NSS is a secret share scheme that computes the secret NSS(s) as a multilevel $SS_n(t_n, r_n)$ computation, where n is the level ($n \in 1,2,3,\ldots$), t_n is the threshold at level n and r_n is the redundancy at level n.

Definition 1: let k and d be positive integers, with $k \leq d$. *k-outof-d threshold scheme* is a method of sharing a secret S among a set of d participants where any k participants can compute the value of the secret, but a group of $k-1$ or fewer cannot do it.

Theorem 1: NSS is a k-outof-d threshold secret share scheme

Proof of Theorem 1: Let consider NSS with n=1, that is NNS without nested levels. This represents the particular case where NSS works as the unmodified SS schema (i.e.,Shamir or RRNS) it implements. Since both the Shamir's algorithm and RRNS are t-outof-d threshold schemes [40][41], also NSS is a t-outof-d threshold secret share scheme. So, if n=1, Theorem 1 is proved. If n=2, the shares required to rebuild the secret S at the first nested level are $(S_1, S_2, \ldots, S_x)$ with $x = t$. Using each S_i as an input to the SS run at the first nested level, the output will be $((S_{1,1}, S_{1,2}, \ldots, S_{1,t_2}), (S_{2,1}, S_{2,2}, \ldots S_{2,t_2}), \ldots, (S_{t_1,1}, S_{t_1,2}, \ldots S_{t_1,t_2}))$, where the total number of shares required to rebuild the secret is $x_2 = t_1 \times t_2$. To determine s, the NSS merge schema is applied to the set $s_1, \ldots, s_x 2$ and s is uniquely determined by any t_2 shares. This is the same model of the NSS with $n = 1$ where the number of shares is x_2 and the threshold is t_2. This means that NSS is a t_2-outof-d threshold secret share scheme and, if n=2, Theorem 1 is satisfied. We can prove in the same way that for any value of n, NSS is a k-outof-d threshold secret share scheme where $k = t_n = t_1 \times t_2 \times \ldots \times t_n$. So, Theorem 1 is always satisfied.

Theorem 2: NSS correctly computes s from $\prod_{i=1}^{n}(t_i + r_i)$ chunks of data.

Proof of Theorem 2: Let consider NSS with $n = 1$, with $t = x > 0$ as the number of shares required for rebuilding s and d the number of drones in the system. The value of redundancy is $r = d - t$. Since the secret S is splitted over all the available drones, the number of shares is $d = t + r$. Let consider n=2. The output of the SS run at the first nested level on a generic secret S is $(S_1, S_2, \ldots, S_{t_1+r_1})$. Using each S_i as an input to the SS run at the first nested level, the output will be $((S_{1,1}, S_{1,2}, \ldots, S_{1,t_2+r_2}), (S_{2,1}, S_{2,2}, \ldots S_{2,t_2+r_2}), \ldots, (S_{t_1+r_1,1}, S_{t_1+r_1,2}, \ldots S_{t_1+r_1,t_2+r_2}))$, where the total number of shares is $x_2 = (t_1 + +r_1) \times (t_2 + r_2)$. We can prove in the same way that for any value of n, the total number of shares is $x_n = (t_1 + +r_1) \times (t_2 + r_2) \times \cdots \times (t_n + r_n) = \prod_{i=1}^{n}(t_i + r_i)$. So, Theorem 2 is satisfied.

From Theorem 1, we have that the number of primaries is provided by the following formula:

$$t_1 \times t_2 \times \cdots \times t_n \tag{7}$$

where t_i is the value of t at the i_{th} level.

When $t_1 = t_2 = \cdots = t_n = t$, the total number of primaries would be :

$$t^n \tag{8}$$

If $r_1, r_2, \ldots r_n$ are greater than 0, the total number of chunks can be calculated using the following formulation:

$$(t_1 + r_1) \times (t_2 + r_2) \times \cdots \times (t_n + r_n) \tag{9}$$

When $t_1 = t_2 = \cdots = t_n = t$ and $r_1 = r_2 = \cdots = r_n = r$ the number of chunks would be:

$$(t + r)^n \tag{10}$$

In order to calculate the NSS redundancy level, it is sufficient to subtract the number of primaries from the total number of chunks, therefore the generic formulation that represents the redundancy level would be:

$$(t_1 + r_1) \times (t_2 + r_2) \times \cdots \times (t_n + r_n) - t_1 \times t_2 \times \ldots t_n \tag{11}$$

Considering $t_1 = t_2 = \cdots = t_n = t$ and $r_1 = r_2 = \cdots = r_n = r$ the value of redundancy is:

$$(t + r)^n - t^n \tag{12}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

calculating the binomial power, the above formula can be written as:

$$t^n + nt^{n-1}r + \ldots + \frac{n!}{k!(n-k)!}t^{n-k}r^k + \ldots + npr^{n-1} + r^n - t^n \quad (13)$$

that corresponds to:

$$nt^{n-1}r + \ldots + \frac{n!}{k!(n-k)!}t^{n-k}r^k + \ldots + ntr^{n-1} + r^n \quad (14)$$

From a computational point of view, performing multi-level SS procedures (NSS) can also result in significant computation overheads. This means that the required time for both splitting and merging tasks depends greatly on t, r, the number of nested levels, and the number of nodes that carry out the computation. For a single node scenario, the execution of an SS algorithm has to be sequentially performed, therefore the time required for splitting and merging data can be calculated with the 15 and 16 formulas, respectively.

$$Ts = Ts_1 + (t_1 + r_1) \times Ts_2 + \cdots + (t_{n-1} + r_{n-1})^{n-1} \times Ts_n \quad (15)$$

$$Tm = t_{n-1} \times Tm_n + \cdots + t_1 \times Tm_2 + Tm_1 \quad (16)$$

When $t_1 = t_2 = \cdots = t_n = t$ and $r_1 = r_2 = \cdots = r_n = r$, the time required for splitting and merging data can be calculated with the following formulas:

$$Ts = Ts_1 + (t + r) \times Ts_2 + \cdots + (t + r)^{n-1} \times Ts_n \quad (17)$$

$$Tm = t \times Tm_n + \cdots + t \times Tm_2 + Tm_1 \quad (18)$$

where Ts is the total time required to perform the 'split' procedures, Ts_i is the time taken to split at the i_th level, t is the number of required shares, r is the redundancy, T_m is the total time required to perform the 'merge' procedures, Tm_i is the time required for merge at the i_th level, and n is the number of nested levels.

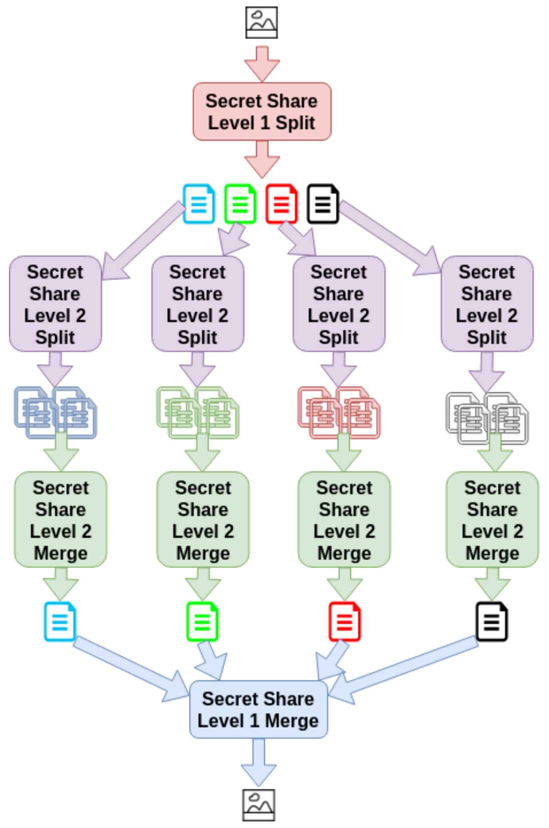

Secret Share Level 1 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Split
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 2 Merge
Secret Share Level 1 Merge

Fig. 4. NSS split and recomposition in a distributed and parallel environment.

In order to reduce the computation time of NSS, a possible solution is to run several SS tasks in parallel. If we could run $(t + r)^{n-1}$ parallel chunks, the time required for splitting and merging tasks can be calculated by means of the 19 and 20 formulas, respectively.

$$Ts = Ts_1 + Ts_2 + \cdots + Ts_n \quad (19)$$

$$Tm = Tm_n + \cdots + Tm_2 + Tm_1 \quad (20)$$

Figure 4 shows an example of 2NSS in a cluster environment. At the top part of the picture, the original data S is first split at the nested level-1. The resulting n chunks are stored in parallel nodes. Furthermore, each node, in turn, splits its chunk into other shares at nested level-2. The merge task behaves similarly. Each merger at level 2 has to combine shares belonging to the same chunk to avoid any errors. As shown using multiple colors in Figure 4, it is possible to combine shares in parallel. The original secret S can be rebuilt after receiving the required chunks provided by the parallel nodes.

6 Implementation

We have implemented our software by using Java programming language. We decided to implement the system in Java to achieve both portability (as we do not need to cross-compile the software for each drone) and performance (as Java is faster than other high-level Object-Oriented Programming Languages such as Python [42]).

In Figure 5 the class diagram of the implemented software is shown.

The software code is composed of ten classes. The main class is *NSS*, and its main purpose is to instantiate other classes. *CameraManager* is a class that takes the control of the camera and provides files to be split. *Splitter* is an abstract class, it is extended from *SplitterRRNS* and *SplitterShamir* to provide the functionality to split a generic file according to RRNS and Shamir's schema, respectively. *CodifierBase64* is a class that takes as input an array of bytes and creates a chunk. To process multiple chunks at the same time, the class *Splitter* extends the class *Thread*. *Chunk* is the class that represents fragments of a picture. It implements the *Serializable* interface to send data over a network. *Sender* is the class that distributes chunks among drones through a socket. *Receiver* is the class that opens a socket and waits for connections from other drones. Finally, *ReceiverThread* is the class to manage the incoming connections with other drones and storing chunks on the drones' memory. In order to handle multiple connections, this class implements the interface *Runnable*.

The workflow for splitting a picture is as the following: the NSS class within the main method instantiates a *CodifierBase64*, a *CameraManager*, and a *Splitter* objects (for example, we created a SpitterRRNS object, the same can be applied for SplitterShamir). The method *takePicture* of *CameraManager* class takes pictures and sends them to the Splitter. The method *run* of the Splitter class splits the picture into chunks and invokes the method *send* in the *Sender* class. Figure 6 shows the sequence diagram for splitting and sending chunks.

The workflow for receiving chunks is as the following: the NSS class within the main method instantiates an object *Receiver*

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

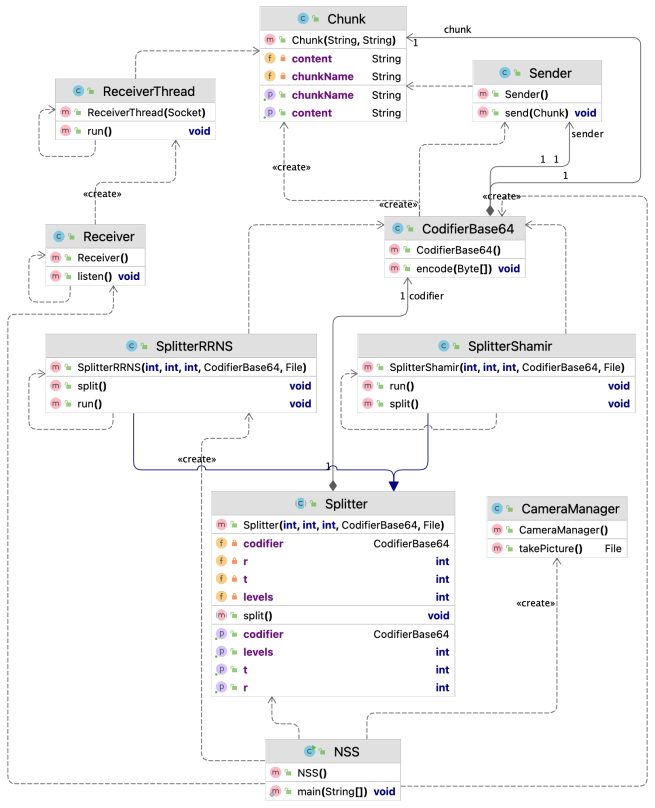


Fig. 5. Class diagram of the developed system.

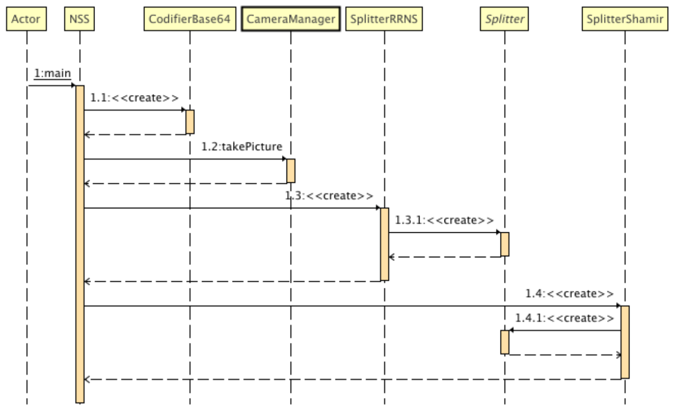


Fig. 6. Sequence diagram for splitting and sending chunks.

that opens a *ServerSocket* to listen to a specific port. When a client connects the ServerSocket, a new *Socket* object is returned. The socket object is passed to the *ReceiverThread* constructor that launches a new Thread for handling the communication with the sender. Within the *run* method, the receiverThread deserializes the incoming Chunk using the *readObject* method. Figure 7 shows the sequence diagram for receiving chunks.

7 Performance Results and Analysis

We performed extensive experiments to measure the performance of our NSS technique in drone fleet photo storage scenarios, ranging from "Top Secret" to "Unclassified". In all cases, we assumed that when a photo is taken by a drone, it is immediately

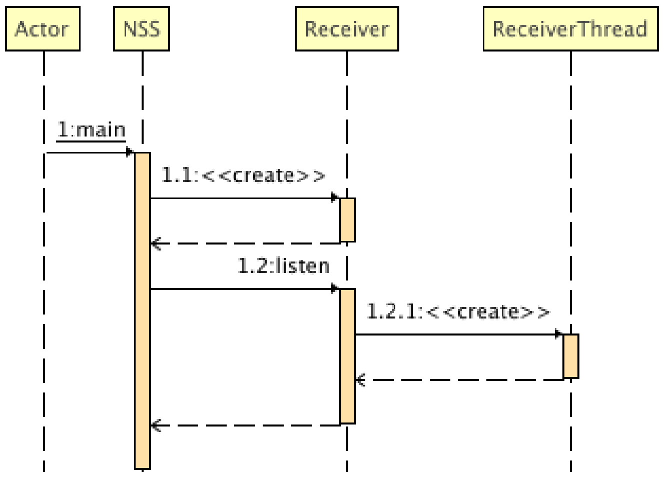


Fig. 7. Sequence diagram for receiving chunks.

split into chunks, through a SS or NSS mechanism, that are spread among other drones of the fleet for storage. Specifically, using the 2NSS technique and two of the major SS algorithms previously analyzed (Shamir's schema and RRNS), we calculated: (a) the security of each algorithm in terms of the percentage of secrets that an enemy could recompose upon stealing several drones; and (b) the applicability of our approach for drones, in terms of execution time and energy consumption for the splitting task. We did not assess the time required for merging photos because we assumed that the merging process is performed at the base using a third-party system, and thus irrelevant for flying drones. We also neglected the network transfer time required to send chunks amongst drones because the size of the photos is generally very small.

As the goal of this scientific work is to verify the applicability of the 2NSS in drone fleets, we only analyzed the splitting tasks as they are supposed to be performed on flying drones. The software has been written in Java, therefore it can be run on any drone with the Java Virtual Machine (JVM) installed. In this paper, we used a Raspberry Pi 3 to simulate hardware capability of drones; this is because it is quite common to install Raspberry Pis on commercial drones like 'DJI Matrice 600' and 'DJI Phantom' to launch attacks [43] or to build Raspberry-based drones [44], [45]. The hardware (HW) and Software (SW) characteristics of drones considered in our experiments are shown in Table 1.

TABLE 1
Hardware and Software characteristics of drones.

Parameter	Value
RAM	1 GB LPDDR2-900 SDRAM
CPU	1.2 GHZ quad-core ARM Cortex A53
Storage	MicroSD 8GB W throughput: 10.9 MB/s; R throughput: 22.6 MB/s

We carried out our analyses considering different drone fleet sizes (10, 100 and 1000 drones) that go on a mission. Furthermore, we assumed that 1000 photos are taken during each mission; for example, 100 photos per drone when the size of a fleet is 10 drones. For each level of SS, we considered the number of p primaries fixed to 5 (because RRNS in our implementation needs at least 5 primaries), whereas we varied the redundancy value for

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

each level r_1 and r_2 from 0 to 7. We remark that the configuration with $r_2 = 0$ is different from a single level SS because the chunks provided as output of level-1 are still processed at level-2. As we discussed in Section 5, the total number of processes can be calculated using Equation 9. The size of the input file, that is a photo or a small video in our case, varied from 0.5 MB to 50 MB. Table 2, summarizes all simulated experimental setups.

TABLE 2
Summary of performed experiments setup.

Parameter	Values
Environment	Drones (see table 1)
Secret Share algorithms	Shamir, RRNS
Tasks	Splitting
Primaries (p)	5
Redundancy (r_1, r_2)	0, 1, 3, 5, 7
Total SS Levels	2
File size [B]	500k, 5M, 50M
Fleet size	10, 100, 1000
Missions	30
Photos for Mission	1000
Mission Classification	Top Secret, Secret, Confidential, Restricted, Unclassified

7.1 2NSS Requirements

For each mission type (i.e., Top Secret, Secret, Confidential, Restricted, Unclassified), we aim at assessing different parameters (i.e., security, execution time, and battery consumption) to identify the optimal configuration of our system. In our definitions, the term "security" refers to the number of secrets that an enemy could recompose by stealing drones. Considering the "Top Secret" mission, due to the sensitivity of a photo, we expect that an enemy could obtain all chunks to recompose all photos only if it can capture all drones of a fleet. With reference to "Unclassified" missions, we expect to recompose a sensitive photo even if a big portion of the fleet is lost.

Top Secret missions usually imply high risks, and therefore the total mission time should be as short as possible to reduce the risk of drones be caught or destroyed. Another solution to reduce such risks for drones could be the use of smaller drones. Due to the constrained dimensions of drones, even the batteries have to be smaller; therefore, for such kind of missions, we expect a very low battery consumption to process each sensitive photo. The only constraint of the energy usage for "Unclassified" missions depends on the fact that drones have a limited battery capacity, and therefore, it is not possible to consume a lot of energy for processing a picture. Table 3 summarizes the features of each mission type according to our requirements. Considering different missions or requirements (for instance providing more redundancy to "Top Secret" data) the requested features could be different.

7.2 Security

In our experiments, three scenarios with fixed fleet sizes (i.e., 10, 100, 1000) are considered. We investigated the retrieval rate of the proposed 2NSS technique compared with a classical single-level SS both the Shamir's Schema and in the RRNS algorithm. The retrieval rate can be seen as an "availability" property (in terms of the number of photos that can be recomposed considering the number of drones that return to base for "Unclassified" missions), as well as the "security" property (in terms of the number of photos

TABLE 3
NSS Requirements.

Mission Classification	% of fleet that is required to recompose all secret	Mission Time	Energy Required
Top Secret	100%	Very Low	Very Low
Secret	80%	Low	Low
Confidential	60%	Moderate	Moderate
Restricted	60%	Moderate	Moderate
Unclassified	40%	No constrain	Moderate

that the enemy is able to recompose when capturing drones in "Top Secret" missions). In particular, we considered different fleets, composed of 10, 100 and 1000 drones, that fly to take photos of a strategic site. The mission ends when the total number of 1000 photos are taken. We made 30 missions and calculated the average number of photos that it is possible to recompose.

In Figure 8 the x axis of each graph is the percentage of drones that return to the base (availability) or are captured (security). The y axis is the percentage of photos that could be recomposed. We remark that to recompose the secret, differently from SS where all the chunks can be combined, in NSS the enemy has to know how to combine chunks, otherwise, in some configurations s/he may need to make up to 1.7 Billion attempts [6].

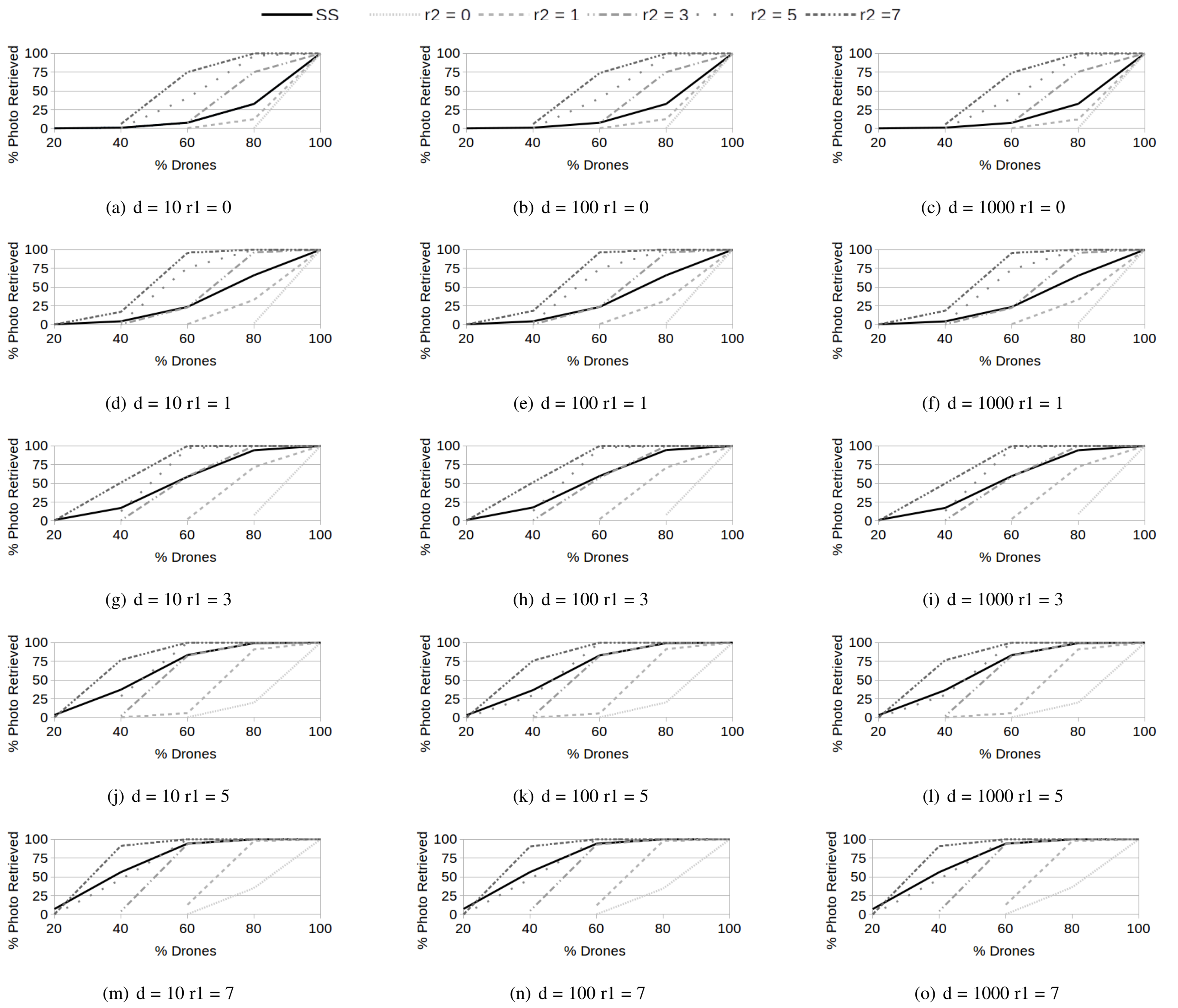
Figures 8(a), 8(d), 8(g), 8(j) and 8(m) refer to a fleet size equal to 10 and r_1 equal to 0,1,3,5,7 respectively. Figures 8(b), 8(e), 8(h), 8(k) and 8(n) refer to the fleet size equal to 100 and r_1 equal to 0,1,3,5,7 respectively. Figures 8(c), 8(f), 8(i), 8(l) and 8(o) refer to the fleet size equal to 1000 and r_1 equal to 0,1,3,5,7 respectively.

In these graphs, the continuous lines represent other SS algorithms, while the dashed lines correspond to 2NSS. It is noteworthy to mention that the percentage of retrieved photos is the same for both Shamir's Schema and RRNS algorithms.

In our experiment, we considered different values of r_1 and r_2 (the redundancy degree of the first and second level of SS, respectively) when comparing all proposed approaches. As we can see, in all configurations the lightest dashed line ($r_2 = 0$) is on the right and under the SS (our reference); this means that to recompose photos, we need (or the enemy needs) a larger portion of the fleet. In other words, losing up to about 20% of the drone fleet size does not allow an enemy to reconstruct any photo (case with $r_1 \leq 3$) of a very low number of pictures ($r_1 > 3$). This configuration can be useful for Top Secret missions because it allows to recompose all the pictures only when the 100% of the fleet is available. If more than 20% of the drone fleet size is compromised, the mission needs to be repeated. Considering $r_2 = 7$, we have the opposite behavior; that is, the darkest dashed line is on the left and above of the SS line. This means that we are (or the enemy is) able to recompose almost all photos with a small portion of the fleet. This kind of configuration is optimal for unclassified missions because retrieving information is more important than disclosing it. Considering other values of r_1 and r_2, we will have intermediate conditions. In our experiments, we considered a random distribution of chunks among drones. Therefore, they could store multiple chunks of a specific picture and no chunks of others. From experimental results, we have seen that it is possible to rebuild almost 50% of the taken pictures by using the 90% of the fleet in the configuration when $r_1 = r_2 = 0$.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115



(a) d = 10 r1 = 0 (b) d = 100 r1 = 0 (c) d = 1000 r1 = 0

(d) d = 10 r1 = 1 (e) d = 100 r1 = 1 (f) d = 1000 r1 = 1

(g) d = 10 r1 = 3 (h) d = 100 r1 = 3 (i) d = 1000 r1 = 3

(j) d = 10 r1 = 5 (k) d = 100 r1 = 5 (l) d = 1000 r1 = 5

(m) d = 10 r1 = 7 (n) d = 100 r1 = 7 (o) d = 1000 r1 = 7

Fig. 8. Percentage of photo reconstructed considering different fleet sizes

7.3 Execution Time

In this section, we investigate the execution time of the proposed solutions for the "splitting" task. Differently from the "retrieval rate", the execution time depends on the selected SS algorithm. As discussed in Section 5, the splitting and merging tasks can be carried out either sequentially or in parallel, depending on the use case. In our reference scenario, we are assuming that the split of photos is done sequentially on drones' hardware. This is because each drone has to process and store the same amount of photos, there would be no free drone to parallelize any computation.

Figures 9(a), 9(b) and 9(c) show the execution time of the RRNS for photos of 500kB, 5MB and 50MB, respectively. As it can be seen, the time required by 2NSS-RRNS for "Unclassified" missions ($r_1 = r_2 = 7$) is roughly 10 times more than the time required by RRNS for the 500 KB file. The difference between the execution time of RRNS and 2NSS-RRNS, decreases by increasing the photo sizes. That is, to split the 5MB photo 2NSS-RRNS requires roughly 7 times of what required by RRNS; and to split a 50MB video, 2NSS-RRNS requires roughly five times of what required by RRNS. This behavior is justifiable, because in our implementation, the execution of the 2NSS-RRNS is executed on smaller photos (about 0.25 times). For "Top Secret" missions ($r_2 = 0$), the gap between SS and NSS is reduced, because as we discussed in Equation 9 of Section 5, the total number of generated chunks depends on the value of the redundancy in each level.

Figures 10(a), 10(b) and 10(c) show the execution time of Shamir's Schema algorithm for files of 500kB, 5MB and 50MB, respectively.

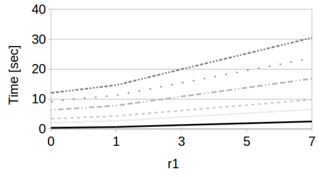
Unlike RRNS, the execution of 2NSS-Shamir's Schema splitting task is done on chunks that are bigger than the input photo; as a result, the execution time of the 2NSS-Shamir's Schema increases faster than 2NSS-RRNS.

7.4 Mission Time

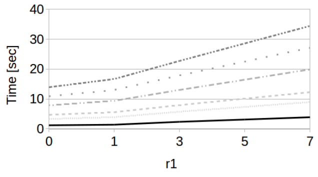
The "mission time" is the time required by the drone fleet to take photos and to share chunks among drones. It should be as small as possible because, during that phase, the drones are closer to the enemy and the probability to be captured is higher. Bigger fleets

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

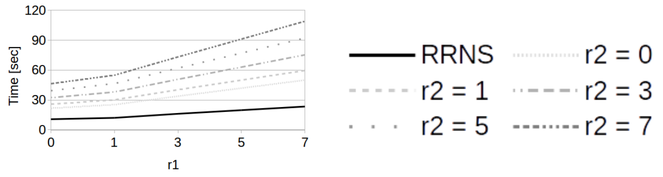
This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115



(a) File Size = 500kB

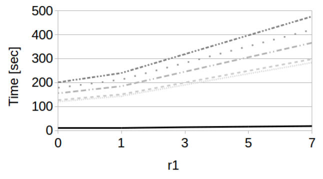


(b) File Size = 5MB

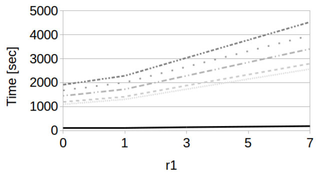


(c) File Size = 50MB

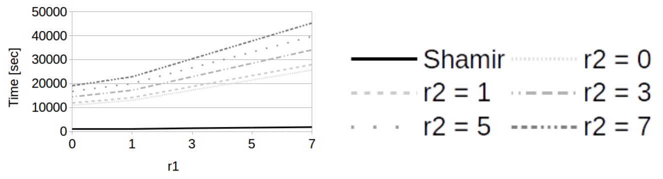
Fig. 9. Execution time of RRNS split for different file sizes. *N.B. the scale of y is different in figure c)*



(a) File Size = 500kB



(b) File Size = 5MB



(c) File Size = 50MB

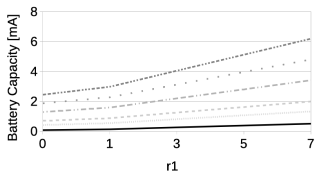
Fig. 10. Execution time of Shamir's schema split for different file sizes. *N.B. the scale of y is different in all figures*

ensure lower mission time because each drone has to process fewer number of photos. For example when taking 1000 photos for each mission, if the fleet is composed of 1000 drones, each of them has to process only one photo, therefore the "mission time" of 2NSS-RRNS and 2NSS-Shamir's Schema are equal to times shown in Figures 9 and 10, respectively. When 50 MB photos are taken, the processing time in the "Top Secret" configuration is less than a minute for 2NSS-RRNS and about 8 hours for 2NSS-Shamir's Schema. For "Unclassified" missions, the time increase up to 100 seconds for 2NSS-RRNS and about 14 hours for 2NSS-Shamir's Schema.

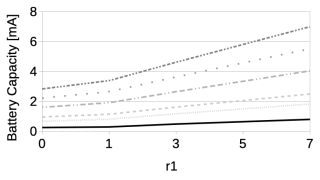
If the fleet is composed of 100 drones the time required is 10 times larger: about 17 minutes for 2NSS-RRNS and 5 days for 2NSS-Shamir's Schema for "Unclassified" missions ($r_1 = r_2 = 7$) and file sizes larger than 50 MB. This value is still acceptable for 2NSS-RRNS, but too high for 2NSS-Shamir's Schema. For fleets with 10 drones, the mission time of 2NSS-RRNS is about 2 hours and 47 minutes for 50 MB photos when $r_1 = r_2 = 7$. For "Unclassified" missions, 2NSS-Shamir's Schema requires more than 7 days.

7.5 Energy Consumption

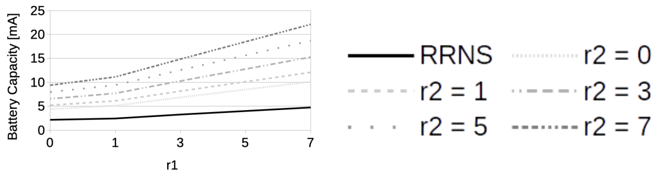
The capacity of a drone's battery is limited, and thus longer mission times require bigger batteries to power on each drone. The goal of this section is the dimensioning of batteries to evaluate the real applicability of NSS techniques. To make our calculation, we used the results from [46] for Raspberry Pi 3 at 400% CPU load. We know that Raspberry Pis require 5V batteries, therefore the instant energy required can be calculated easily from the benchmark [46] and is equal to $1mJ/s$.



(a) File Size = 500kB



(b) File Size = 5MB



(c) File Size = 50MB

Fig. 11. Battery consumption time of RRNS split for different file sizes. *N.B. the scale of y is different in figure c)*

Figure 11 shows the required current to split a single photo using RRNS (for the single-level version as well as the nested one). In particular, Figures 11(a), 11(b) and 11(c) show the behavior for Low-resolution Photos, High-resolutions photos, and videos respectively.

For the splitting time, the energy consumption also depends on the fleet size. The best case is the scenario with 1000 drones (shown in Figure 11) because each drone has to process only one photo or video. As it can be seen, the configuration with lower degrees of redundancy requires less current (only a few mA), therefore, also in this case, this type of configuration matches the requirements defined for "Top Secret" missions. Considering higher values of redundancy, the current required increases accordingly; in the worst case scenario (File size= 50MB $r_1 = r_2 = 7$), to process a single video about 23 mA are needed, which is an acceptable value for drones. As discussed for mission time, decreasing the drone fleet size increases the capacity of the battery required to power drones.

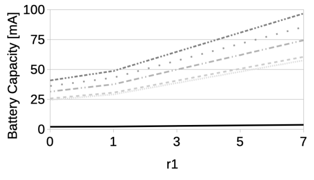
In Figures 12(a), 12(b) and 12(c) the energy required for processing a single photo with Shamir's Schema is shown. The energy required from this algorithm, to process a picture, is very high as compared with RRNS, especially for larger photos/videos. Therefore, it is not possible to use these techniques on drones to process photos/videos.

7.6 Discussion

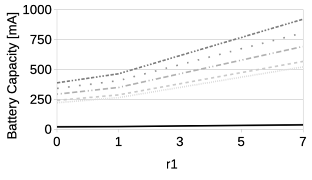
In the following, we summarize our findings when comparing NSS techniques with regard to their retrieval rate, execution time, and energy consumption in relation to the file size, algorithm, redundancy degree, etc.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

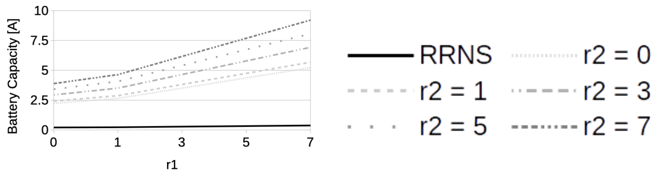
This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115



(a) File Size = 500kB



(b) File Size = 5MB



(c) File Size = 50MB

Fig. 12. Battery consumption time of Shamir's schema split for different file sizes. *N.B. the scale of* y *is different in all figures*

Experiments showned that the retrieval rate does not depend on the adopted SS algorithm. In particular, we have seen that by using lower values of *r*, the enemy has to steal a greater number of drones to recompose all the taken pictures. This configuration, considering the requirements defined in Section 7.1, is optimal for "Top Secret" missions.

Execution time and energy consumption depend on the adopted SS technique. With regard to the execution time, we have considered the time required for splitting photos. We have seen that the overhead introduced from the 2NSS-RRNS decreases as the size of the input file increases. 2NSS-Shamir behaves differently. The splitting time depends on the energy consumption and the capacity of batteries to power on drones. For each photo, 2NSS-RRNS requires a few mAs, whereas 2NSS-Shamir's Schema requires almost 10A. This significant difference makes the 2NSS-Shamir's Schema approach not adequate for splitting and sharing sensitive photos among drones.

Considering all these aspects, starting from requirements defined in Table 3, we created Table 4 that shows the most suitable configuration for each mission type.

TABLE 4
Summary of the most suitable configuration for each mission type.

Mission Classification	**Feet size**	**Algorithm**	r_1	r_2
Top Secret	1000	RRNS	0	0
Secret	100	RRNS	1	1
Confidential	100	RRNS	3	3
Restricted	100	RRNS	5	5
Unclassified	10	RRNS	7	7

8 Conclusions and Future Work

In this paper, we discussed the use of NSS techniques for splitting and storing photos among a fleet of drones. We introduced the NSS technique to overcome limitations of SS algorithms to be deployed on drones, mostly because the security level of these algorithms is often fixed, and thus impossible to re-configure them based on the sensitivity of a mission. Specifically, we compared a 2NSS technique with two of the major SS algorithms (Shamir's Schema and RRNS), considering three aspects (retrieval rate, execution time, and power consumption). Experiments showed that lower redundancy values are good for "Top Secret" missions, because, the enemy needs to capture all drones of a fleet to recompose all photos. We also observed that the 2NSS-Shamir's Schema, due to its complexity, is not suitable for drones. In future works, we plan to implement data deduplication techniques to optimize the picture transfer time among drones and to combine NSS with private-public key algorithms.

Acknowledgment

This work has been partially supported by the Department for drug policies of the Presidency of the Council of Ministers (Italian Government) through the project InstradaME (CUP F49I20000100001); the National Operational Programme Metropolitan Cities 2014-2020(PON METRO) through the project MeSm@rt (CUP F41I18000230006) and the Knowledge Foundation of Sweden through the HITS Profile.

References

[1] M. Raffaele, M. T. Caccamo, G. Castorina, S. Lanza, S. Magazù, G. Munaò, and G. Randazzo, "Designing drones by combining finite element and atomistic simulations: a didactic approach," *Atti della Accademia Peloritana dei Pericolanti - Classe di Scienze Fisiche, Matematiche e Naturali*, vol. 99, no. S1, p. 33, 2021.
[2] "Civil drones in the European Union." http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/571305/EPRS_BRI%282015%29571305_EN.pdf. Last access: June 2022.
[3] "Iran shoots down US drone aircraft, raising tensions further in Strait of Hormuz." https://edition.cnn.com/2019/06/20/middleeast/iran-drone-claim-hnk-intl/index.html. Last access: June 2022.
[4] "Dutch police will stop using drone-hunting eagles since they weren't doing what they're told." https://www.theverge.com/2017/12/12/16767000/police-netherlands-eagles-rogue-drones. Last access: June 2022.
[5] "Executive Order 12356–National security information." https://www.archives.gov/federal-register/codification/executive-order/12356.html#part1. Last access: June 2022.
[6] A. Galletta, J. Taheri, M. Fazio, A. Celesti, and M. Villari, "Overcoming security limitations of secret share techniques: the nested secret share," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 289–296, 2021.
[7] Y. Ke, M. Zhang, X. Zhang, J. Liu, T. Su, and X. Yang, "A reversible data hiding scheme in encrypted domain for secret image sharing based on chinese remainder theorem," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 2469–2481, 2022.
[8] A. Celesti, A. Galletta, M. Fazio, and M. Villari, "Towards hybrid multi-cloud storage systems: Understanding how to perform data transfer," *Big Data Research*, vol. 16, pp. 1–17, 2019.
[9] A. Galletta, J. Taheri, and M. Villari, "On the applicability of secret share algorithms for saving data on iot, edge and cloud devices," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 14–21, July 2019.
[10] T. Noguchi, M. Nakagawa, M. Yoshida, and A. G. Ramonet, "A secure secret key-sharing system for resource-constrained iot devices using mqtt," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pp. 147–153, 2022.
[11] L. Bertojo and W. Puech, "Correction of secret images reconstructed from noised shared images," in *2022 Eleventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–6, 2022.
[12] A. Galletta, M. Fazio, A. Celesti, and M. Villari, "Verifiable secret share for file storage with cheater identification," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 788–793, 2020.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

[13] Y.-q. Fang, J.-b. Liao, and L.-y. Lai, "Verifiable secret sharing scheme using merkle tree," in *2020 International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pp. 1–4, 2020.
[14] A. K. Chattopadhyay, D. Ghosh, P. Maitra, A. Nag, and H. N. Saha, "A verifiable (n, n) secret image sharing scheme using xor operations," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pp. 1025–1031, Nov 2018.
[15] S. Zhang, X. Yong, M. Luo, D. He, and K.-K. R. Choo, "Dssp: Efficient dual-server secret sharing protocol based on password authentication for cloud storage services," *IEEE Systems Journal*, vol. 16, no. 2, pp. 2172–2182, 2022.
[16] V. V. Panchbhai and S. W. Varade, "Hybrid approach to enhance security and friendliness of visual secret sharing scheme," in *2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22)*, pp. 1–6, 2022.
[17] P. Gupta, Y. Li, S. Mehrotra, N. Panwar, S. Sharma, and S. Almanee, "Obscure: Information-theoretically secure, oblivious, and verifiable aggregation queries on secret-shared outsourced data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 34, pp. 843–864, feb 2022.
[18] M. Z. Ejaz, K. Khurshid, Z. Abbas, M. A. Aizaz, and A. Nawaz, "A novel image encoding and communication technique of b/w images for iot, robotics and drones using (15, 11) reed solomon scheme," in *2018 Advances in Science and Engineering Technology International Conferences (ASET)*, pp. 1–6, Feb 2018.
[19] G. Devi Ramaraj, S. Venkatakrishnan, G. Balasubramanian, and S. Sridhar, "Aerial surveillance of public areas with autonomous track and follow using image processing," in *2017 International Conference on Computer and Drone Applications (IConDA)*, pp. 92–95, Nov 2017.
[20] P. Agarwal and M. K. Singh, "A multipurpose drone for water sampling video surveillance," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, pp. 1–5, Feb 2019.
[21] S. Yong, A. L. W. Chung, W. K. Yeap, and P. Sallis, "Motion detection using drone's vision," in *2017 Asia Modelling Symposium (AMS)*, pp. 108–112, Dec 2017.
[22] A. Chowdhery and M. Chiang, "Model predictive compression for drone video analytics," in *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pp. 1–5, June 2018.
[23] K. Akkaya, V. Baboolal, N. Saputro, S. Uluagac, and H. Menouar, "Privacy-preserving control of video transmissions for drone-based intelligent transportation systems," in *2019 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–7, 2019.
[24] J. Greenfield and C. Thaxton, "Data acquisition and logging system for a drone-mounted atmospheric sensor suite," in *SoutheastCon 2022*, pp. 178–183, 2022.
[25] S. Alzahrani, Y. Xiao, and W. Sun, "An analysis of conti ransomware leaked source codes," *IEEE Access*, vol. 10, pp. 100178–100193, 2022.
[26] A. Vehabovic, N. Ghani, E. Bou-Harb, J. Crichigno, and A. Yayimli, "Ransomware detection and classification strategies," in *2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 316–324, 2022.
[27] K. K. Soni and A. Rasool, "Cryptographic attack possibilities over rsa algorithm through classical and quantum computation," in *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 11–15, 2018.
[28] V. Bhatia and K. Ramkumar, "An efficient quantum computing technique for cracking rsa using shor's algorithm," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, pp. 89–94, 2020.
[29] M. M. Kaidenko and S. O. Kravchuk, "Anti-jamming system for small unmanned aerial vehicles," in *2021 IEEE 6th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD)*, pp. 1–4, 2021.
[30] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022.
[31] J. Peng, Z. Zhang, Q. Wu, and B. Zhang, "Anti-jamming communications in uav swarms: A reinforcement learning approach," *IEEE Access*, vol. 7, pp. 180532–180543, 2019.
[32] Y.-H. Chou, G.-J. Zeng, X.-Y. Chen, and S.-Y. Kuo, "Multiparty weighted threshold quantum secret sharing based on the chinese remainder theorem to share quantum information," *Scientific Reports*, vol. 11, no. 1, p. 6093, 2021.
[33] A. Galletta, A. Celesti, F. Tusa, M. Fazio, P. Bramanti, and M. Villari, "Big mri data dissemination and retrieval in a multi-cloud hospital storage system," in *Proceedings of the 2017 International Conference on Digital Health*, pp. 162–166, ACM, 2017.
[34] A. Galletta, L. Bonanno, A. Celesti, S. Marino, P. Bramanti, and M. Villari, "An approach to share mri data over the cloud preserving patients' privacy," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 94–99, IEEE, 2017.
[35] W. Othman, M. Fuyou, K. Xue, and A. Hawbani, "Physically secure lightweight and privacy-preserving message authentication protocol for vanet in smart city," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12902–12917, 2021.
[36] A. Anand and A. K. Singh, "Secret sharing based watermarking for copy-protection and ownership control of medical image," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 01–07, 2021.
[37] A. Galletta, M. Fazio, A. Celesti, and M. Villari, "On the applicability of secret share algorithms for osmotic computing," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.
[38] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, Nov. 1979.
[39] R. W. Watson and C. W. Hastings, "Self-checked computation using residue arithmetic," *Proceedings of the IEEE*, vol. 54, pp. 1920–1931, Dec 1966.
[40] P.-Y. Lin and C.-S. Chan, "Invertible secret image sharing with steganography," *Pattern Recognition Letters*, vol. 31, no. 13, pp. 1887–1893, 2010.
[41] M. G. Babenko, A. Tchernykh, E. Golimblevskaia, N. V. Hung, and V. Chaurasiya, "Computationally secure threshold secret sharing scheme with minimal redundancy.," in *ICCS-DE*, pp. 23–32, 2020.
[42] "A Performance Comparison Between C, Java, and Python." https://medium.com/swlh/a-performance-comparison-between-c-java-and-python-df3890545f6d. Last access: Jan 2023.
[43] "How Wi-Fi spy drones snooped on financial firm." https://www.theregister.com/2022/10/12/drone-roof-attack/. Last access: Dec 2022.
[44] "Raspberry Pi Drone: How to Build Your Own." https://all3dp.com/2/raspbery-pi-drone-simply-explained/. Last access: Dec 2022.
[45] "How to build a Raspberry Pi drone." https://www.thedronegirl.com/2021/04/05/how-to-build-a-raspberry-pi-drone/. Last access: Dec 2022.
[46] "Power Consumption Benchmarks." https://www.pidramble.com/wiki/benchmarks/power-consumption. Last access: June 2022.

Antonino Galletta is an assistant professor at the University of Messina. In 2020 he received a PhD from the University of Reggio Calabria (Italy). He received his B.Sc. and M.Sc. (with honours) in Computer Engineering from the University of Messina. Currently, he is the PI of "InstradaME" an Italian project funded by the Ministry of Interior. His main research activities focus on the security of Cloud/Edge/IoT technologies for Smart cities and eHealth solutions, including Big Data management and Blockchain. Due to his contribution to the research, he has been selected among 200 top young researchers in Mathematics and Computer Science to participate in the prestigious "Heidelberg Laureate Forum" held in Heidelberg in September 2019. He has been the winner of two editions of the precious "Leonardo Innovation Award" in 2018 (First Placed) and 2017 (Second Placed). He was the guest editor of several SIs on Q1 journals. Currently, he is Associate Editor of Elsevier SusCom Journal; one of the members of the Technical Committee of the Elsevier Computer Communication Journal and a reviewer of more than 15 respected Journals, Chair of several International Conferences and co-author of more than 50 manuscripts.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3263115

Javid Taheri is a Full Professor in the Department of Computer Science at Karlstad University, Sweden. He received his PhD in Mobile Computing from the University of Sydney (Australia) in 2007, and his Bachelor's and Master's of Electrical Engineering from Sharif University of Technology, Tehran (Iran) in 1998 and 2000, respectively. He is the recipient of many awards including being selected as one of the top 200 young researchers in the world by the Heidelberg Forum in 2013, the recipient of several best paper awards since 2007, and the recipient of the prestigious IEEE Middle Career Researcher award from TSCS in Scalable Computing in 2019. His research interests include Cloud Computing, Edge/Fog Computing, Network Function Virtualization, Software-defined Networking, and AI-based optimisation techniques. He is the editor of a book on Big Data and Software-Defined Networks, editor of another book on Edge Computing, and the lead author of a book on Edge Intelligence. He coauthored > 200 scientific articles and papers, has served as an editor for > 20 journals, and is a member of the organizing team for > 40 international conferences.

Antonio Celesti is an associate professor at the University of Messina. He received the PhD in Advanced Technology for Information Engineering in 2012 at the University of Messina (Italy). He worked as a collaborator in many National and International Projects.From 2014, he is Assistant Researcher at University of Messina (Italy) working on the EU FP7 project FRONTIERCITIES - European Cities driving the Future Internet. He was Adjunct Professor of Cloud Computing (2013- 2014), Parallel and Distributed Programming (2013-2014), Foundations of Computer Science (2012-2013, 2013-2014), Computer Science Skills (2012-2013, 2013-2014), Security of Systems (2011- 2012, 2012, 2013), Open Source and Grid Computing (2012-2013), Information Systems and ICT (2011-2012). His main research interests include service composition, big data, system federation, and security.

Maria Fazio is an Assistant Professor in Computer Science at the University of Messina (Italy). She participated in several national and international projects, such as the EU-FP7 RESERVOIR (2008 - 2011), EU-FP7 VISION Cloud (2010-2013), EU-FP7 CloudWave (2013-16), EU-H2020 BEACON (2015-17), EU-H2020 URBANITE (2020-2023). She is one of the members of the Editorial Review Board of several international journals (e.g., Scientific Reports, Computing-Springer-Verlag Wien, Wireless Communications and Mobile Computing-John Wiley & Sons and Hindaw) and has been co-editor of a number of Special Issues published in international journals (e.g., IEEE Cloud Computing, International Journal of Distributed Systems and Technologies (IJDST)-IGI Global). Her research interests are focused on computing continuum, with particular reference to intelligent microservice orchestration, auto-configuring mesh networks at the Edge, and security in IoT-Edge ecosystems.

Massimo Villari is a Full Professor in Computer Science at the University of Messina (Italy). He is actively working as IT Security and Distributed Systems Analyst in Cloud Computing, virtualization and Storage and is one of the creators of the Osmotic Computing Paradigm. He was Scientific ICT Responsible for the EU Project frontierCities, the Accelerator of FIWARE on Smart Cities – Smart Mobility. He is strongly involved in EU Future Internet initiatives, specifically Cloud Computing and Security in Distributed Systems. He is co-author of more than 200 scientific publications and patents in Cloud Computing (Cloud Federation), Distributed Systems, Wireless Network, Network Security, Cloud Security and Cloud, Edge and IoTs, and recently in Osmotic Computing and AI. He is a Co-Founder of UniME Spin-Off Alma Digit S.R.L since 2017. He is the General Chair of IEEE CCGRID in 2022 in Taormina (Messina - Sicily) and the General Chair of IEEE ISCC2022. Currently, he is the Head of Computer Science School and Rector delegate on ICT for the whole University of Messina and Academic Consultant for the Messina Municipality in the Area of Smart Cities. He is in the Top 2% Most Influential Scientists in Stanford University 2021 List, in the area of Analysis of Artificial Intelligence & Image Processing. He is the Head of the Research Group FCRLab in UniME with more than 25 young researchers.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/