The VLDB Journal manuscript No.
(will be inserted by the editor)

Secret Sharing for Cloud Data Security

A Survey

Varunya Attasena · Jérôme Darmont · Nouria Harbi

Received: date / Accepted: date

Abstract Cloud computing helps reduce costs, increase business agility and deploy solutions with a high return on investment for many types of applications. However, data security is of premium importance to many users and often restrains their adoption of cloud technologies. Various approaches, i.e., data encryption, anonymization, replication and verification, help enforce different facets of data security. Secret sharing is a particularly interesting cryptographic technique. Its most advanced variants indeed simultaneously enforce data privacy, availability and integrity, while allowing computation on encrypted data. The aim of this paper is thus to wholly survey secret sharing schemes with respect to data security, data access and costs in the pay-as-you-go paradigm.

Keywords Cloud computing · Secret sharing · Data privacy · Data availability · Data integrity · Data access

Varunya Attasena
The computer engineering department, Engineering faculty at Kamphaeng Saen, Kasetsart University Kamphaeng Saen Campus, Nakhon Pathom, 73140, Thailand
Tel.: +66-34-281-074 #7528
Fax: +66-99-695-415
E-mail: fengvry@ku.ac.th

Jérôme Darmont
Université de Lyon, Lyon 2, ERIC EA 3083, 5 avenue Pierre Mendès France, 69676 Bron Cedex, France
Tel.: +33-478-774-403
Fax: +33-478-772-375
E-mail: jerome.darmont@univ-lyon2.fr

Nouria Harbi
Université de Lyon, Lyon 2, ERIC EA 3083, 5 avenue Pierre Mendès France, 69676 Bron Cedex, France
Tel.: +33-478-774-492
Fax: +33-478-772-375
E-mail: nouria.harbi@univ-lyon2.fr

1 Introduction

Cloud computing is currently booming, with companies of all sizes adopting associated technologies to benefit from resource and cost elasticity. However, data security remains one of the top concerns for cloud users and would-be users. Security issues, both inherited from classical distributed architectures and specific to the new framework of the cloud, are indeed numerous, especially at the data storage level of public clouds [22].

Critical security concerns in cloud storage are depicted in Figure 1, which highlights the major issues in cloud data security, i.e., data privacy, availability and integrity. In particular, cloud architectures might not be sufficiently safeguarded from inside attacks. In virtual environments, a malicious user might be able to break into "neighboring" virtual machines located on the same hardware, and then steal, modify or delete the other users' data [57,2,29,56,104,102,47]. In such environments, users are indeed usually granted with superuser access for managing their virtual machines. A malicious superuser can access real network components and thus launch attacks [2,11]. Moreover, virtualization allows the rollback of a virtual machine to some previous state if necessary. Although this rollback feature provides flexibility to the users, it can also revert the virtual machine to previous security policies and configuration control [2,47]. Eventually, virtual machine migration is run to improve quality of service. During such migration processes, which typically do not shut down services, virtual machine contents are exposed to the network, and problems such as network transfer bottlenecks and data damage may occur [2,47,97].

Classical data security approaches, i.e., data encryption [9,39], data anonymization [24], replication [69], data verification [87], data separation [96,68,103] and

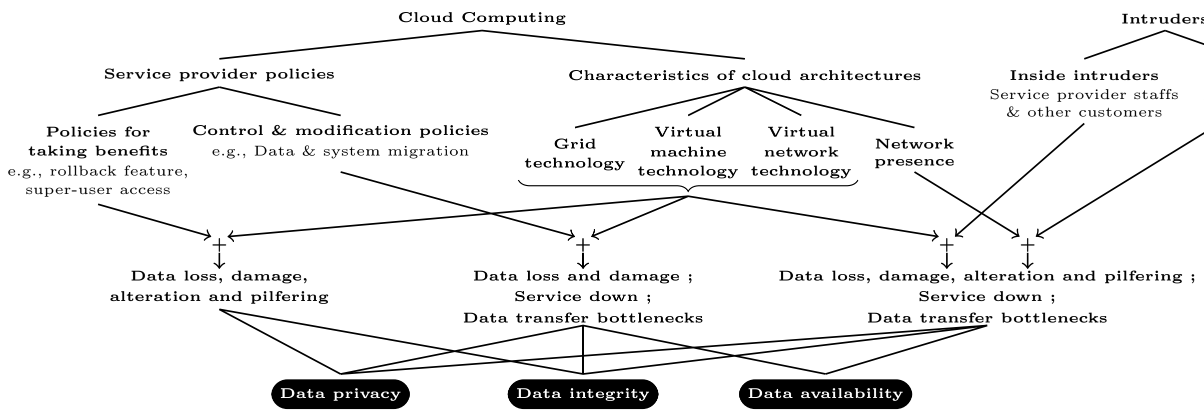

Cloud Computing
Service provider policies
Characteristics of cloud architectures
Intruders
Inside intruders
Service provider staffs & other customers
Outside intruders
Policies for taking benefits
e.g., rollback feature, super-user access
Control & modification policies
e.g., Data & system migration
Grid technology
Virtual machine technology
Virtual network technology
Network presence
Data loss, damage, alteration and pilfering
Data loss and damage ; Service down ; Data transfer bottlenecks
Data loss, damage, alteration and pilfering ; Service down ; Data transfer bottlenecks
Data privacy
Data integrity
Data availability

Fig. 1: Data security issues in the cloud

differential privacy [34], can solve most data security issues within cloud computing environments (Figure 2), but usually one at a time. Many data-centric cloud applications do not only require data to be secure, but also efficiently accessed, sometimes through complex, analytical queries akin to on-line analysis processing (OLAP) operations. With users seeking to reduce costs in the cloud's pay-as-you-go pricing model, achieving the best tradeoff between data security and access power and efficiency is a great challenge [22,81].

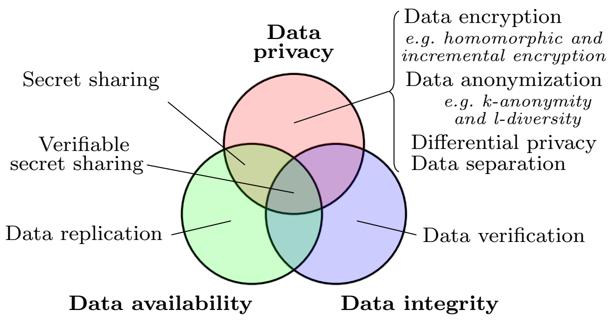


Fig. 2: Features of data security approaches

Existing surveys about distributed data security list security services in distributed storage: authentication and authorization, availability, confidentiality and integrity, key sharing and management, auditing and intrusion detection, and finally useability, manageability and performance [58,29]. Then, network file systems, cryptographic file systems and storage intrusion detection systems are discussed and compared. This pre-cloud review is complemented by a thorough comparison of storage-centric data protection (i.e., network storage devices) in user-centric data protection systems (i.e., cryptographic storage systems and cloud-based storage) [93,29]. Finally, [91,29] provide a short overview of what should be done in terms of data auditing and encryption in the cloud.

Although these surveys do mention secret sharing, they provide few details about this particular cryptographic technique, which was simultaneously introduced by Shamir [76] and Blakley [12] in 1979 and can be particularly useful nowadays in the context of cloud computing, e.g., to safely manage and analyze big data. Threshold secret sharing schemes indeed transform sensitive data into individually meaningless data pieces (called shares) that are distributed to n participants akin to CSPs. Computations can then be performed onto shares, but yield meaningless individual results. The global result can only be reconstructed knowing individual results from several participants (more than threshold $t \leq n$). Moreover, some secret sharing variants simultaneously enforce data privacy, availability and integrity, which no other security scheme achieves. Eventually, secret sharing can be used by both CSPs, with data being shared within their cloud infrastructure, and users, who can dispatch sensitive data over several providers. Since some secret sharing schemes also support homomorphism, they allow data analysis on shares, thus allowing data access cost optimization.

To the best of our knowledge, secret sharing schemes (SSSs) up to 2008 have only been surveyed with respect to bounds on share size and global data volume [8]. In this paper, we also include the most recent SSSs and complement [8] by analyzing the objectives of each SSS, the security and data analysis features a user can expect, and the costs implied in a cloud computing environment.

The remainder of this paper is organized as follows (Figure 3). Section 2 describes the principles of secret sharing and classifies SSSs into eleven groups, whose properties are thoroughly detailed. SSSs in a given group are also positioned with respect to one an-

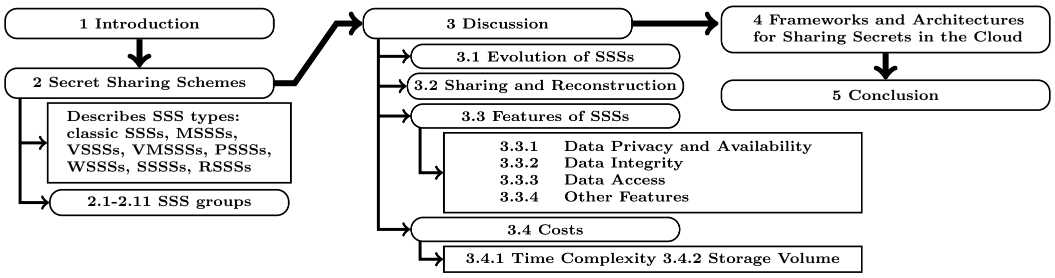
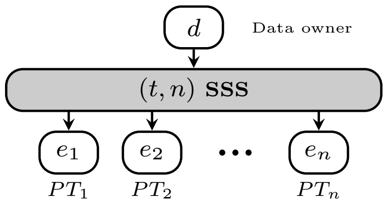
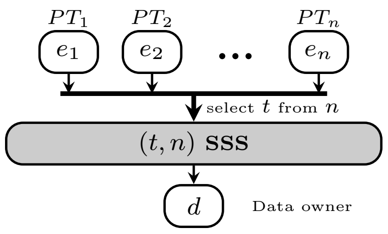

1 Introduction
2 Secret Sharing Schemes
Describes SSS types: classic SSSs, MSSSs, VSSSs, VMSSSs, PSSSs, WSSSs, SSSSs, RSSSs
2.1-2.11 SSS groups
3 Discussion
3.1 Evolution of SSSs
3.2 Sharing and Reconstruction
3.3 Features of SSSs
3.3.1 Data Privacy and Availability
3.3.2 Data Integrity
3.3.3 Data Access
3.3.4 Other Features
3.4 Costs
3.4.1 Time Complexity 3.4.2 Storage Volume
4 Frameworks and Architectures for Sharing Secrets in the Cloud
5 Conclusion

Fig. 3: Schematic map of the paper

other. In Section 3, we compare all surveyed SSSs with respect to data security, queries over shares, and storage and computing costs. Moreover, we present SSS-based frameworks that provide secure storage, e.g., databases or data warehouses, in the cloud in Section 4. Finally, Section 5 concludes this paper, recaps open research issues and describes sample applications in the cloud.



(a) Sharing process



(b) Reconstruction process

Fig. 4: Classical secret sharing

2 Secret Sharing Schemes

The threshold SSSs we survey in this paper are primarily aimed at enforcing privacy. Individual secret d is divided into n so-called shares $\{e_i\}_{i=0,\cdots,n}$, each share e_i being stored by a different participant (PT) PT_i (Figure 4(a)). Each share e_i is meaningless to PT_i. A subset of $t \leq n$ PTs is required to reconstruct the secret (Figure 4(b)). Thence, a convenient side effect of SSSs is data availability, since up to $n - t$ PTs may disappear without preventing secret reconstruction. Classical SSSs [76,12,3,49,54,71,46,72,61] mainly differ in sharing methods, which bear different security properties with different data storage and CPU requirements.

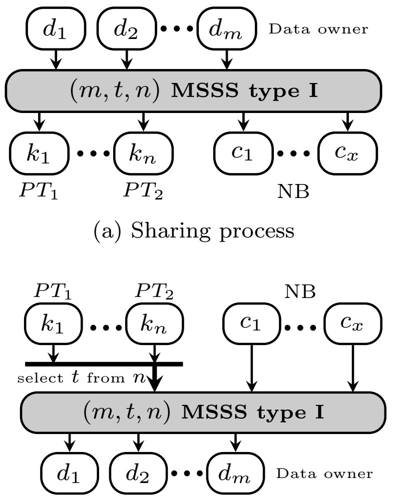
A major drawback of initial SSSs is the multiplication of the initial data volume by the number of PTs. Multi secret sharing schemes (MSSSs) thus aim to reduce computation, storage and data transfer costs by sharing and reconstructing more than one secret at once. Some MSSSs achieve an overall shared data volume (i.e., at all PTs') that is close to that of original secret data. We categorize MSSSs into two types.

In MSSSs type I [94,90], data are shared with the help of keys. m secrets $\{d_j\}_{j=1,\cdots,m}$ and n keys $\{k_i\}_{i=1,\cdots,n}$ are used to construct x shares $\{c_h\}_{h=1,\cdots,x}$, where $m \leq x$. Shares are stored in a news bulletin board (NB), whereas each key k_i is stored at PT_i (Figure 5(a)). To reconstruct the m secrets, all or some (depending on the MSSS) shares and t keys are used (Figure 5(b)).

In MSSSs type II [18,75,61,83], m secrets $\{d_j\}_{j=1,\cdots,m}$ are divided into n shares $\{e_i\}_{i=1,\cdots,n}$, where $m \leq t \leq n$. In case $m > t$, secrets are first organized into blocks that are fewer than t. Then, each block is divided into n shares at once. Finally each share e_i is stored by PT_i (Figure 6(a)). As in SSSs, reconstructing the secrets requires t PTs (Figure 6(b)).

SSSs and MSSSs assume that all players, i.e., PTs and NB, are honest and always provide valid information (data and keys). However, in reality, they might not, intentionally or not. Thus, verifiable secret sharing schemes (VSSSs) [73,23,77,53,98] and verifiable multi secret sharing schemes (VMSSSs) [37,99,27,28,89,38, 20,60,52,82,19,78,21,25,16,6,5,80] verify the correctness of data and/or keys before or after reconstruction. Therefore, VSSSs and VMSSSs enforce data integrity in addition to privacy and availability.

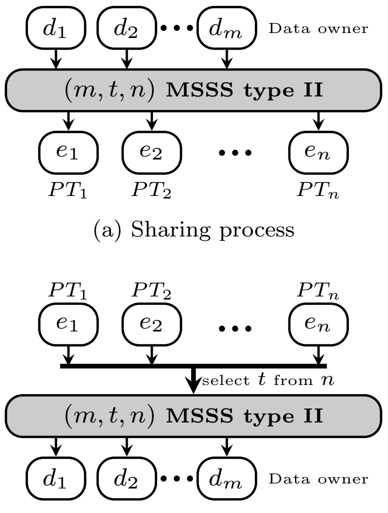
Eventually, some SSSs aim at specic goals. Proactive secret sharing schemes (PSSSs) are based on classical SSSs or VSSSs but, in addition, periodically refresh shares [51,17,92,101,7,31,62]. Refreshing consists in generating a random number at each PT's and sharing it at all other PTs' to modify existing shares. ln most PSSSs [51,92,7,31,62,62], refreshing is synchronous, i.e., shares cannot be reconstructed during the process,



(a) Sharing process

(b) Reconstruction process

Fig. 5: Multiple secret sharing type I



(a) Sharing process

(b) Reconstruction process

Fig. 6: Multiple secret sharing type II

but there are also asynchronous refreshing protocols [17,101] that allow share reconstruction at all times. All PSSSs include a share verification process that verifies whether shares are up-to-date when refreshing. If shares are obsolete, they may be safely deleted or recovered from other shares. Since shares are periodically refreshed even if secrets have not been updated, an intruder has little time to compromise them.

However, the refreshing process in PSSSs induces extra costs, i.e., computing costs for periodically sharing random numbers among PTs and modifying shares (time complexity is $O(mn^2)$ [51]); and high communication costs for commuting PTs with each other, whose cost is at least n times that of sharing secrets. Because of these costs, and since PSSSs reuse the data sharing and reconstruction processes of the classical SSSs that are detailed in this section, we do not survey PSSSs further.

Weighted secret sharing schemes (WSSSs) extend classical SSSs by introducing a priority among PTs by assigning to each PT a weight, i.e., the number of shares it stores [10,64,45,33]. More precisely, in these schemes, any secret d is divided into w shares such that $w \geq n$. Each PT$_i$ holds w_i shares such that $w = \sum_{i=1}^{n} w_i$. If $n = w$ or $w_i = 1\ \forall i$, we fall back to a classical SSS. Secret reconstruction is only possible by a group of PTs holding at least t shares, with $w_i < t \leq w\ \forall i$. One single PT cannot reconstruct the secret, since $w_i < t\ \forall i$.

Social secret sharing schemes (SSSSs) extend from WSSSs by allowing weights to be adjusted depending on the situation, e.g., if some PTs are found insincere [67, 66,100,65] . Even though WSSSs and SSSSs bring in a more flexible PT management, they induce a higher share volume, i.e., at least n times the original data volume *vs.* at most n times for previous SSSs, supposing that individual shares use up the same volume as secrets. Thus, we do not survey them further.

Finally, function secret sharing schemes (FSSSs) [14, 59,15] aim at protecting data transfers over networks when keyword search is performed on outsourced, replicated data. A function f is shared into n functions $f_1, \cdots, f_n$ such that $f = \sum_{i=1}^{n} f_i$. Each function f_i is associated with a data node akin to a participant PT$_i$ in classical secret sharing. When the user issues a search query with some keyword k, $f_i(k)$ is sent to PT$_i$ $\forall i = 1 \cdots n$. Then data at each PT$_i$ are matched with $f_i(k)$. The local result R_i is shared as $f_i(R_i)$ and sent back to the user, who can finally reconstruct a global result with $t \leq n$ values of $f_i(R_i)$. However, FSSSs do not fit in our data outsourcing scenario since data are replicated in clear form. Thus, we do not survey them further. Yet, FSSSs are quite recent and hybridizing them with other SSSs surveyed in this section could help solve this issue.

We categorize SSSs into eleven groups (Table 1) with respect to their basic type, i.e., SSSs and MSSSs types I and II, as well as eventual data or key verification. We survey all groups in the following subsections. Moreover, we introduce the parameters and notations used throughout this section in Table 2.

2.1 Group 1: Classical Secret Sharing Schemes

The very first (t,n) SSS [76] enforces data security by using a random polynomial (Equation 1). This polynomial is generated over a finite field such that coefficient c_0 is the secret and other coefficients $c_{u=1,\cdots,t-1}$ are ran-

Table 1: Classification of secret sharing schemes

		SSSs	MSSSs type I	MSSSs type II	
Verification	**None**	**Group 1** [76, 12, 3] [49, 54, 46] [71, 72] [61] SSS	**Group 2** [94, 90]	**Group 3** [18, 75, 83] [61] MSSS	**SSSs/MSSSs**
	Data	**Group 4** [73, 23, 77]	**Group 7** [37]	**Group 10** [6, 5]	**VSSSs/VMSSSs**
	Keys	**Group 5** [53]	**Group 8** [99, 27, 28] [89, 38, 20] [60],[52]-I&II		
	Data & keys	**Group 6** [98]	**Group 9** [82, 19, 78] [21, 25, 16]	**Group 11** [80]	

dom integers. Then, each share e_i is created by Equation 2 and stored at PT_i. A number $t \leq n$ of PTs can reconstruct the original polynomial by Lagrange interpolation over a finite field, which enforces data availability even if $n - t$ PTs fail. A sample application of this scheme is given in Figure 7, where $t = 4$ and $n = 6$. The random polynomial of degree $t - 1 = 3$ is $e_i = f(i) = i^3 - 5i^2 + 2i + 4$, where 4 is the secret. The six shares $\{(i, e_i)\}_{i=1,\cdots,6}$ (plotted in blue) are (1,2), (2,-4), (3,-8), (4,-4), (5,14) and (6,52).

$$f(i) = \sum_{u=0}^{t-1} c_u \times i^u \tag{1}$$

$$e_i = f(i) \tag{2}$$

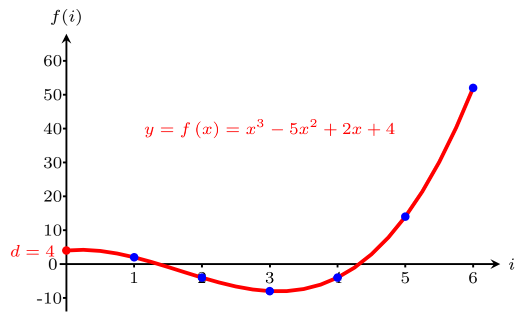
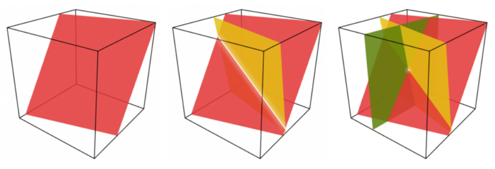


Fig. 7: Secret sharing by polynomial interpolation

In Blakley's SSS [12], each PT is associated with an hyperplane in a t-dimensional space over a finite field. Hyperplanes, i.e., shares, intersect in a point that is the secret, which can be reconstructed by solving the hyperplanes' equation system. A sample application of this scheme is given in Figure 8, where $t = 2$ and $n = 3$ (there are thus three hyperplans).

Table 2: Secret sharing schemes' parameters

Parameter	**Definition**
m	Number of secrets
D	Secret data such that $D = \{d_1, \cdots, d_m\}$ and $D = \{b_1, \cdots, b_o\}$
d	Secret in integer format
$\|\|d\|\|$	Storage size of d
d_j	j^{th} element of D in integer format
n	Number of PTs
t	Number of shares necessary for reconstructing the secret
γ	Number of PTs of the first group in [83]
PT_i	PT number i
ID_i	Identifier of PT_i
g	Number of groups of PTs
G_r	r^{th} group of PTs such that $G_r \subseteq \{PT_i\}_{i=1,\cdots,n}$ and $G_r = \{PT_{r,1}, \cdots, PT_{r,g}\}$
$PT_{r,v}$	PT number v of G_r
$ID_{r,v}$	Identifier of $PT_{r,v}$ of G_r
o	Number of data blocks
b_l	l^{th} block of D such that $b_l = \{d_{l,1}, \cdots, d_{l,t}\}$ with fixed-sized blocks and $b_l = \{d_{l,1}, \cdots, d_{l,t_l}\}$ with variable-sized blocks
t_l	Number of shares necessary for reconstructing the secret in b_l (in case of variable-sized blocks)
$d_{l,q}$	q^{th} element of b_l in integer format
e_i	Share stored at PT_i
$e_{j,i}$	j^{th} share stored at PT_i
$e_{l,i}$	Share of b_l stored at PT_i
c_h	h^{th} share stored at the NB
$c_{j,h}$	h^{th} share of d_j stored at the NB
$c_{l,h}$	h^{th} share of b_l stored at the NB
$c_{l,q,h}$	h^{th} share of $d_{l,q}$ in b_l stored at the NB
$c_{r,l,h}$	h^{th} share of b_l from G_r stored at the NB
k_i	Key stored at PT_i
$k_{i,q}$	Key number q stored at PT_i
$k_{r,i}$	Key stored at PT_i of G_r
$\|\|k\|\|$	Storage size of keys
s_d_i	Signature stored at PT_i
s_d_l	Signature of d_l
$s\_d_{l,i}$	Signature of b_l stored at PT_i
$s\_d_{l,q}$	Signature number q of b_l
s_k_i	Signature of PT_i's key
$s\_k_{r,v}$	Signature of $PT_{r,v}$'s key of G_r
$\|\|s\|\|$	Storage size of signatures
$p, p_1, p_2 \ldots$	Big prime numbers
$A, A_1, A_2 \ldots$	Matrices
$f, f_1, f_2 \ldots$	Functions
$H, H_1, H_2 \ldots$	Hash functions



from: http://en.wikipedia.org/wiki/Secret_sharing

Fig. 8: Secret sharing through hyperplan intersection

[3] exploits the Chinese remainder theorem [30]. First, $n+1$ uniquely relatively primes[1] $\{p_i\}_{i=0,\cdots,n}$ are determined such that $p_0 < p_1 < \cdots < p_n$ and $\prod_{i=1}^{t} p_i > p_0 \prod_{i=1}^{t-1} p_{n-i+1}$. Then, n shares $\{e_l\}_{l=1,\cdots,n}$ are created by Equations 3 and 4, where u is a random positive integer. Finally, secret d is reconstructed from t shares by Equations 5 and 6.

$$e_i = y \bmod p_i \tag{3}$$
$$y = d + u \times p_0 \tag{4}$$
$$d = y \bmod p_0 \tag{5}$$
$$y \equiv e_i \bmod m_i \tag{6}$$

All subsequent SSSs extend the three foundation schemes above. [54] extends from [3] to reduce the size of shares. Moreover, this SSS can reconstruct a secret from t or more shares, whereas previous schemes exploit exactly t shares. In the sharing process, the secret is split in t. Share creation from the t splits and secret reconstruction proceed as in [3]. All other SSSs seek to improve polynomial interpolation.

[71] proceeds in two steps. First, secret d is divided into t intermediate shares $\{u_v\}_{v=1,\cdots,t}$ by mapping d to the x-axis of a random polynomial. Second, these t shares are divided again into n actual shares $\{e_i\}_{i=1,\cdots,t}$ by Equation 7, where A_1 is an $n \times t$ random matrix. Secret d is reconstructed from a polynomial of degree t created by Equation 8. $\{u_v\}_{v=1,\cdots,t}$ are reconstructed by Equation 9, where A_2 is a $t \times t$ inverse matrix seeded from t rows of matrix A_1.

$$[e_1, \cdots, e_n]^T = A_1 \times [u_1, \cdots, u_t]^T \tag{7}$$
$$\prod_{a=1}^{t} (x - u_a) \equiv 0 \bmod p \tag{8}$$
$$[u_1, \cdots, u_t]^T = A_2 \times [e_1, \cdots, e_t]^T \tag{9}$$

The second step enforces availability and is optional. A sample application of the first step is given in Figure 9, where $d = 10$ and $t = 3$. The polynomial equation of degree 3 $(x-u_1)(x-u_2)(x-u_3) \equiv x^3 - 21x^2 + x - 10 \equiv 0 \bmod 31$ is created with the help of prime $p = 31$ and random positive integers $u_1 = 19$, $u_2 = 22$ and $u_3 = 11$, where u_1, u_2, u_3 match with condition $u_3 \equiv d \times (u_1 \times u_2) \bmod p$.

[1] Uniquely relatively primes are random prime numbers that are related to each other by some conditions.

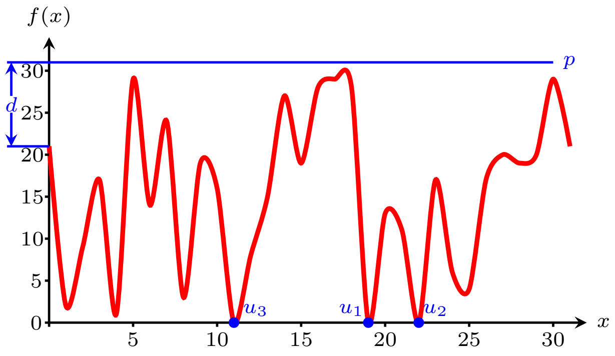


Fig. 9: [71]'s secret mapping step

In [72], a secret d is split into $t-1$ smaller data units $\{u_v\}_{v=1,\cdots,t-1}$ to reduce global share volume. Then, a polynomial equation of degree $t-1$ is created by running recursive functions $t-1$ times (Equation 10, where y is a random integer) to improve security. Next, n shares $\{e_i\}_{i=1,\cdots,n}$ are created by Equation 11. Finally, data are reconstructed through $t-1$ steps by Lagrange interpolation.

$$f_v(x) = \begin{cases} u_v + y \times x & \text{if } v = 1 \\ u_v + \sum_{w=1}^{v} f_{v-1}(w) \times x^w & \text{otherwise} \end{cases} \tag{10}$$
$$e_i = f_{t-1}(i) \tag{11}$$

[49] extends from [76] to guarantee the t-consistency of shares, i.e., any subset of t shares or more always reconstruct the same secret. A random polynomial function $f(x)$ is created as in [76] $(f(0) = d)$. However, $k_{i,1}$ and $k_{i,2}$ are random keys stored at PT_i and $k_{i,2}$ is do not need to be distinct from each other. Next, n shares $\{c_i\}_{i=1,\cdots,n}$ are created by Equation 12 and stored on the NB. Secret d can be reconstructed by Lagrange interpolation from t pairs $\{k_{i,1}, c_i + k_{i,2}\}$.

$$c_i = f(k_{i,1}) - k_{i,2} \tag{12}$$

[46,61] extend from [73] (Section 2.4). However, none verifies the correctness of shares. In addition, both approaches verify a strong t-consistency property. The verification processes guarantee that any subset of t shares or more (created by summing n random polynomial functions of degree $t-1$ in [73]) always reconstruct the same secret, but that any subset of t shares or fewer cannot. Verification time is slower in [46] than in [61].

Eventually, in (t, L, n) threshold ramp SSSs (RSSSs) introduced by Blackley [13], the secret cannot be reconstructed from $t - L$ or less shares (vs. $t - 1$ or less in above SSSs), with $1 \leq \ell \leq L-1$ shares being allowed to leak information about the secret. Thus, RSSSs propose a tradeoff between security and efficiency (measured by entropy) [55]. Let $H(d)$ and $H(e_i)_{i=1,\cdots,n}$ be the entropy of the secret and its shares, respectively. In SSSs, $H(e_i) \geq H(d)$, while in RSSSs, $H(e_i) = H(d) \div L$. [55] also introduces the notion of strong and weak RSSSs, and shows that Shamir-based SSSs may be weak. Yet, most of the following RSSSs still extend Shamir's SSS.

2.2 Group 2: Multi Secret Sharing Schemes Type I

The first (m, t, n) MSSS type I [94] extends from [76] to reduce share volume and execution time. All secrets are shared at once, with share volume being controlled to remain close to that of secrets. To share m secrets $\{d_j\}_{j=1,\cdots,m}$ among n PTs, n keys $\{k_i\}_{i=1,\cdots,n}$ are created with a two-variables one-way function. Then, a polynomial (Equation 1) is created over a finite field [76], with a degree $w = \max(m, t) - 1$.

Moreover, coefficients $\{u_j\}_{j=1,\cdots,m}$ are secrets $\{d_j\}_{j=1,\cdots,m}$ and other coefficients $\{u_j\}_{j=(m+1),\cdots,t}$ are random integers. Next, $m+n-t$ shares $\{c_h\}_{h=1,\cdots,(m+n-t)}$ are generated by Equation 13 and are published on a NB. Finally, secrets are reconstructed by Lagrange interpolation from t or more keys and w shares.

$$c_h = \begin{cases} f(k_h) & \text{if } 1 \leq h \leq n \\ f(H(h)) & \text{if } n+1 \leq h \leq n+m-t \end{cases} \tag{13}$$

[90] extends from [80] (Section 2.11) by reducing execution time and dynamically adjusting data block size. In the sharing process, secrets are organized into o unfixed size blocks $\{b_l\}_{l=1,\cdots,o}$. Data block b_l stores t_l secrets $\{d_{l,q}\}_{q=1,\cdots,t_l}$. Next, keys k_i are randomly selected and matrix $A = [a_{x,y}]_{n\times\max(t_1,\cdots,t_o)}$ is created by Equation 14, where $l = 1, \cdots, o$, $u_{l,q}$ is a random integer and $A_l = [a_{x,y}]_{n\times t_l}$ such that A_l is made of the first t_l columns of A. Next, $o \times t_l$ shares $\{c_{l,h}\}_{l=1,\cdots,o;h=1,\cdots,t_l}$ are created by Equation 15, where v is a random integer. Finally, each key k_i is shared at PT_i and $\{c_{l,h}\}_{l=1,\cdots,o;\, h=1,\cdots,t_l}$, A and t_l are published on the NB. In the reconstruction process, $\{u_{l,q}\}_{q=1,\cdots,t_l}$ is created by solving Equation 14. Then, secrets are reconstructed by solving Equation 15.

$$[f_l(k_1), \cdots, f_l(k_n)]^T = A_l \times [u_{l,1}, \cdots, u_{l,t_l}]^T \tag{14}$$

$$\begin{aligned} c_{l,h} = &\textstyle\sum_{q=1}^{t_l} d_{l,q} \times v^{(q-1)(\sum_{l=1}^{h-1} t_l + h-1)} \\ &+ \textstyle\sum_{q=1}^{t_l} u_{l,q} \times v^{(t_l+q-1)(\sum_{l=1}^{h-1} t_l + h-1t)} \end{aligned} \tag{15}$$

2.3 Group 3: Multi Secret Sharing Schemes Type II

The first (m, t_m, n) MSSS type II [18] extends [76] to share m secrets with different threshold access structures. In the sharing process, PT identifiers $\{ID_i\}_{i=1,\cdots,n}$ are randomly chosen from distinct integers. With respect to secret d_j, t_j and a prime p_j are selected such that $t_1 \leq t_2 \leq \cdots \leq t_m$, $p_1 < p_2 < \cdots < p_m$, $P = \prod_{j=1}^{m} p_j$ and $d_j < p_j$. Next, a polynomial of degree $t_m - 1$ (Equation 16) is created with coefficients $\{u_v\}_{v=1,\cdots,t-1}$ being integers chosen by the Chinese remainder theorem [30] and the uniqueness theorem of interpolating polynomial. $\forall_v \in [0, t-1], u_v \equiv u_{j,v} \bmod p_j \forall j = 1, \cdots, m$ where $u_{j,v}$ is a coefficient of a random polynomial function of degree $t_j - 1$ ($f_j(x) = \sum_{w=0}^{t_j-1} u_{j,v} \times x^w$ [76]) and $u_{j,0} = d_j$. Shares $\{e_i\}_{i=1,\cdots,n}$ are generated by equation 17. Finally, ID_i and e_i are stored at PT_i, whereas $\{t_j\}_{j=1,\cdots,m}$ and $\{p_j\}_{j=1,\cdots,m}$ are retained at the user's. Secret d_j is reconstructed from p_j and t_j pairs (ID_i, e_i) by equations 18 and 19.

$$f(x) = \sum_{v=0}^{t_m-1} u_v \times x^v \tag{16}$$

$$e_i = f(ID_i) \bmod P \tag{17}$$

$$f_j(0) \equiv d_j \bmod p_j \tag{18}$$

$$f_j(x) \equiv f(x) \bmod p_j \tag{19}$$

[75] shares unfixed sized data blocks with a linear equation. There are t_l secrets $\{d_{l,q}\}_{q=1,\cdots,t_l}$ in block b_l ($t_{l_1} < t_{l_2}$ if $l_1 < l_2$). Then, $o \times n$ shares $\{e_{l,i}\}_{l=1,\cdots,o;\, i=1,\cdots,n}$ are created by multiplying b_l with random matrix $A = [a_{x,y}]_{n\times\max(t_1,\cdots,t_o)}$ by Equation 20, where $A_l = [a_{x,y}]_{n\times t_l}$ and A_l is built from the first t_l columns of A. Next, o shares $\{e_{l,i}\}_{l=1,\cdots,o}$ are stored at PT_i and matrix A is published on the NB. Finally, secrets from block b_l are reconstructed from matrix A_l and t_l shares $\{e_{l,i}\}_{i=1,\cdots,t_l}$ by solving linear Equation 20.

$$[e_{l,1}, \cdots, e_{l,n}]^T = A_l \times [b_l]^T \tag{20}$$

[61]'s MSSS extends from [61]'s SSS (Section 2.1) with a new sharing process. At PT_i, shares $\{u_{i,a,j}\}_{a=1,\cdots,n}$ of secrets are computed and distributed to other PTs [73] (Section 2.4). However, PT_i's actual share $e_{i,j}$ is computed by weighting the sum of other PTs' shares (Equation 21), where w_a is a random integer (weight).

$$e_{i,j} = \sum_{a=1}^{n} w_a \times u_{i,a,j} \tag{21}$$

In [83], PTs are categorized into two groups: $G_1 = \{PT_i\}_{i=1,\cdots,\gamma}$ and $G_2 = \{PT_i\}_{i=\gamma+1,\cdots,n}$, with the objective of reducing share volume. PTs of G_1 store only one key and one share. To share m secrets $\{d_j\}_{j=1,\cdots,m}$, a key k_i and an identifier ID_i are defined for each PT_i. Next, a first polynomial $f_1(x)$ is defined by Equation 22, where coefficients $\{u_{1,v}\}_{v=1,\cdots,t-1}$ are random integers. Then, n shares $\{e_{1,i}\}_{i=1,\cdots,n}$ are created by Equation 23. Moreover, $(m-1)\times\gamma$ pseudo shares $\{e_{j,i}\}_{j=2,\cdots,m;i=1,\cdots,\gamma}$ are generated with a pseudo-random number generator, keys $\{k_i\}_{i=1,\cdots,\gamma}$ and shares $\{e_{1,i}\}_{i=1,\cdots,\gamma}$. Next, $m-1$ polynomials $f_2(x),\cdots,f_m(x)$ (Equation 22) are solved from $m\times\gamma$ pseudo shares $\{e_{j,i}\}_{j=2,\cdots,m;i=1,\cdots,\gamma}$ and m secrets $\{d_j\}_{j=2,\cdots,m}$ to construct the other $(m-1)\times(n-\gamma)$ shares $\{e_{j,i}\}_{j=2,\cdots,m;i=\gamma+1,\cdots,n}$ (Equations 22 and 23). Eventually, each $PT_i \in G_1$ stores k_i and one share $e_{1,i}$; and each $PT_i \in G_2$ stores shares $\{e_{j,i}\}_{j=1,\cdots,m}$.

$$f_j(x) = d_j + \sum_{v=1}^{t-1} u_{j,v} \times x^v \tag{22}$$

$$e_{j,i} = f_j(ID_i) \tag{23}$$

To reconstruct the secrets, t of n PTs in both G_1 and G_2 are selected. If $PT_i \in G_1$, pseudo shares $\{e_{j,i}\}_{j=2,\cdots,m}$ are generated as above. Then, secret data are reconstructed by Lagrange interpolation from their shares, $m\times t$ pseudo shares and t IDs.

2.4 Group 4: Data-Verifiable Secret Sharing Schemes

There are only three (t,n) VSSSs in this group. [73] helps each PT verify other PTs' shares with the help of an RSA cryptosystem [74]. To share secret d at PT_i, a random polynomial function f_i (Equation 24) is created such that $d=\sum_{i=1}^{n} w_{i,0}$. Then, t signatures $\{s\_d_{i,v}\}_{v=0,\cdots,t-1}$ are created (Equation 25, where p is a prime and $d=\log_p\prod_{i=1}^{n} y_i$) and shared on the NB. Then, shares $\{u_{i,a}\}_{a=1,\cdots,n}$ are created by Equation 26 and distributed to other PTs. PT_i's actual share e_i is created by summing other PTs' shares (Equation 27) if they are correct (Equation 28). Secrets are reconstructed by Lagrange interpolation.

$$f_i(x) = \sum_{v=0}^{t-1} w_{i,v} \times x^v \tag{24}$$

$$s\_d_{i,v} = \begin{cases} y_i & \text{if } v=0 \\ p^{w_{i,v}} & \text{otherwise} \end{cases} \tag{25}$$

$$u_{i,a} = f_i(a) \tag{26}$$

$$e_i = \sum_{a=1}^{n} u_{a,i} \tag{27}$$

$$p^{u_{a,i}} = \prod_{v=0}^{t-1} (s\_d_{a,v})^{i^v} \tag{28}$$

[23] extends from [76] by verifying the correctness of reconstructed secrets. To this aim, in the sharing process, a signature s_d is created for each secret d (Equation 29, where u is a random integer). Then, s_d is published on the NB.

$$s\_d = u^d \bmod p \tag{29}$$

In the reconstruction process, secret d is reconstructed from t shares by secure multi-party computation (SMC) [95] (Equation 30). Next, a multi-prover zero-knowledge argument [85] helps verify correctness. Secret d is correct if $u^{v_1''+\cdots+v_n''} \times s\_d^{v_0} = \prod_{i=1}^{n} v_i' \bmod p$, where $\{v_i'\}_{i=1,\cdots,n}$ and $\{v_i''\}_{i=1,\cdots,n}$ are generated by Equations 31 and 32, respectively, and $\{v_i\}_{i=0,\cdots,n}$ and $\{w_i\}_{i=0,\cdots,n}$ are random integers such that $d=\sum_{i=1}^{n} w_i$.

$$d = \sum_{i\in G}\left(e_i \times \prod_{j\in G, j\neq i} j/(j-i)\right) \tag{30}$$

$$v_i' = u^{v_i} \bmod p \tag{31}$$

$$v_i'' = v_i - v_0 \times w_i \bmod p \tag{32}$$

[77] exploits NTRU encryption [74] and a hash function to verify the correctness of shares. First, n pairs of PT_i keys $(k_{i,1},k_{i,2})_{i=1,\cdots,n}$ are randomly created with NTRU. Then, shares e_i and signatures s_d_i are created by Equations 33 and 34, respectively, where $\{x_i\}_{i=1,\cdots,n}$ are random integers, w is a random polynomial called blinding value and f is a random polynomial [76]. Keys $(k_{i,1},k_{i,2})$ and shares e_i are stored at PT_i and $\{x_i\}_{i=1,\cdots,n}$ and signatures $\{s\_d_i\}_{i=1,\cdots,n}$ are published on the NB. Before reconstruction, each share e_i is verified for correctness by Equations 35 and 36. Finally, t pairs of $(e_i,x_i)_{i=1,\cdots,n}$ help reconstruct secrets from the polynomial by Lagrange interpolation.

$$e_i \equiv (w \times k_{i,1} + f(x_i)) \bmod p_1 \tag{33}$$

$$s\_d_i \equiv (w \times k_{i,1} + H(f(x_i))) \bmod p_1 \tag{34}$$

$$y_i \equiv k_{i,2} \times e_i \bmod p_1 \bmod p_2 \tag{35}$$

$$y_i \equiv k_{i,2} \times s\_d_i \bmod p_1 \bmod p_2 \tag{36}$$

2.5 Group 5: Key-Verifiable Secret Sharing Schemes

In [53], the only (t,n) VSSS in this group, PT keys and signatures are independent. Hence, if some PTs come or go, the keys of other PTs do not change. PT keys $\{k_i\}_{i=1,\cdots,n}$ and identifiers $\{ID_i\}_{i=1..n}$ are randomly chosen. On the other hand, key signatures $\{s\_k_i\}_{i=1,\cdots,n}$ are generated with the help of an RSA cryptosystem

(Equation 37). Then, key k_i is stored at PT_i, while identifiers and key signatures $(ID_i, s\_k_i)_{i=1,\cdots,n}$ are published on the NB. For sharing secret d, several groups of PTs $\{G_r\}_{r=1,\cdots,g}$ are selected, and then shares $\{e_r\}_{r=1,\cdots,g}$ are created by Equations 38, 39, 40 and 41, where u is a random integer, $p_4 > p_3$ and $p_4 > p_2 > p_1$. Next, v, w, $\{G_r\}_{r=1,\cdots,g}$ and $\{e_r\}_{r=1,\cdots,g}$ are published on the NB. Before reconstruction, the key signature of $PT_i \in G_r$ is verified to check whether $s\_k_i = v^{k_i} \bmod p_2$. If this is true, secrets are reconstructed by Equations 42 and 43.

$$s\_k_i=(p_1)^{k_i} \bmod p_2 \quad (37)$$

$$v=(p_1)^{u} \bmod p_2 \quad (38)$$

$$u \times p_3=a \bmod \phi\,(p_2) \quad (39)$$

$$w_r = d \oplus (s\_k_{r,1})^u \bmod p_2 \oplus \cdots \oplus (s\_k_{r,g})^u \bmod p_2 \quad (40)$$

$$e_r = w_r \times \prod_{x=1}^{t} \frac{1-ID_{r,x}}{-ID_{r,x}} + \sum_{x=1}^{t} \frac{(s\_k_{r,x})^u \bmod p_2 \times \prod_{y=1,y\neq x}^{t} \frac{1-ID_{r,y}}{ID_{r,x}-ID_{r,y}}}{ID_{r,x}} \bmod p_4 \quad (41)$$

$$d = w_r \oplus \left(v^{k_{r,1}} \bmod p_2\right) \oplus \cdots \oplus \left(v^{k_{r,g}} \bmod p_2\right) \quad (42)$$

$$w_r = e_r \times \prod_{x=1}^{t} \frac{-ID_{r,x}}{1-ID_{r,x}} + \sum_{x=1}^{t} \frac{v^{k_{r,x}} \bmod p_2 \times \prod_{y=1,y\neq x}^{t} \frac{-ID_{r,y}}{ID_{r,x}-ID_{r,y}}}{ID_{r,x}-1} \bmod p_4 \quad (43)$$

2.6 Group 6: Key *and* data-verifiable secret sharing schemes

Unlike other schemes, [98]'s (t, n) VSSS verifies the correctness of both keys and shares. Moreover, it achieves a smaller share size than that of secrets, by splitting secrets before the sharing process. In the sharing process, key k_0 and keys $\{k_i\}_{i=1,\cdots,n}$ are randomly selected from a prime and distinct positive integers, respectively. Key signatures $\{s\_k_i\}_{i=0,\cdots,n}$ are constructed by Equation 44, where z is a positive integer and $\varphi(p)$ is Euler's totient function [48]. Next, any secret d is split into t^2 smaller pieces stored in Matrix $D = [d_{x,y}]_{t\times t}$. Then, two types of shares are created (PTs' shares and NB's shares). PTs' shares $\{E_i = \{e_{i,0}, \cdots, e_{i,a}\}\}_{i=1,\cdots,n}$ are sets of randomly distinct positive integers such that $e_{i,0}$ is the sum of all entries in E_i ($e_{i,0} = \sum_{h=1}^{a} e_{i,h}$) and $e_{i,0} < p$. To construct the NB's shares $\{c_i\}_{i=1,\cdots,n}$, polynomial function $f(x)$ (Equation 45) is created from split secrets and PTs' shares by Equations 46 and 47, where A is a Jordan normal form of D^2. Finally, NB's shares $\{c_i\}_{i=1,\cdots,n}$ are constructed from Equations 48 and 49; and share signatures $\{s\_d_{i,j}\}_{i=1,\cdots,n;j=1,\cdots,m}$ are created from Equation 50. Keys k_i and shares E_i are stored at PT_i; shares $\{c_i\}_{i=1,\cdots,n}$, share signatures $\{s\_d_{i,j}\}_{j=1,\cdots,n;j=1,\cdots,m}$, key k_0, key signatures $\{s\_k_i\}_{i=0,\cdots,n}$, p and A are published on the NB.

$$s\_k_i=\begin{cases} k_0^{-1} \bmod \varphi(p) & \text{if } i=0 \\ z^{k_i} \bmod p & \text{if } 1 \le i \le n \end{cases} \quad (44)$$

$$f\,(x)=\sum_{i=1}^{t-1} u_i \times x^{i-1} \quad (45)$$

$$u_i=(((z)^{k_0})^{e_{i,0}})^{-1} y_i \bmod p \quad (46)$$

$$D \times [y_1, \cdots, y_t]^T=[y_1, \cdots, y_t]^T \times A \quad (47)$$

$$c_i=f(v_i) \quad (48)$$

$$v_i=((z)^{k_0})^{k_i} \bmod p \quad (49)$$

$$s\_d_{i,j}=z^{e_{i,j}} \bmod p \quad (50)$$

In the reconstruction process, key k_i is correct if $((z)^{k_i})^{s\_k_{n+1}} = s\_k_i \bmod p$. PT_i's share $e_{i,j}$ is correct if $(((z)^{k_0})^{e_{i,j}})^{s\_k_{n+1}} = s\_d_{i,j} \bmod p$. Next, polynomial function $f(x)$ is reconstructed from t pairs of key and NB's share $\{k_i, c_i\}$ by Lagrange interpolation and Equation 49. Then, $\{y_a\}_{a=1,\cdots,t}$ are created by Equation 51. Finally, secret d is reconstructed by solving Equation 47.

$$y_i = u_i \prod_{j=1}^{a} ((z)^{k_0})^{e_{i,j}} \quad (51)$$

2.7 Group 7: Data-Verifiable Multi Secret Sharing Schemes Type I

The only (m, t, n) VMSSS type I in this group shares and reconstructs all secrets at once with the help of a cellular automaton, to enhance computation performance. Moreover, the correctness of shares is verified before reconstruction [37]. In the sharing process, a set of integers $\left\{u_1, \cdots, u_{\max(m,t)}, \cdots, u_{w+n}\right\}$ is created, where w is a random integer such that $w \ge \max(m, t)$, $u_j = d_j$ if $1 \le j \le \min(t, m)$ and u_j is a random integer when $m < j \le t$. Others values of u_j are created with the help of the cellular automaton. Then, shares $\{c_h\}_{h=1,\cdots,m-t}$ are generated by Equation 52. Shares $\{e_i\}_{i=1,\cdots,n}$ and their signatures $\{s\_d_i\}_{i=1,\cdots,n}$ are created by Equations 53 and 54, where v is a random integer. Finally, each share e_i is shared at PT_i and shares $\{c_h\}_{h=1,\cdots,m-t}$ and signatures $\{s\_d_h\}_{h=1,\cdots,n}$ are published on the NB.

$$c_h=d_{t+h} + u_{t+h}\,(\mathrm{mod}2) \quad (52)$$

$$e_i=u_{m+i} \quad (53)$$

$$s\_d_i=v^{e_i} \bmod p \quad (54)$$

[2] A is a Jordan normal form of D if $DY = YA$, where Y is a row matrix and A is a square, upper triangular matrix whose entries are all the same integer values on the diagonal, all 1 on the entries immediately above the diagonal, and 0 elsewhere.

Before reconstruction, share integrity is verified by Equation 54. Next, $\{u_1, \cdots, u_{\max(m,t)}, \cdots, u_{w+n}\}$ are reconstructed from t shares with the cellular automaton. Finally, all secrets are regenerated by Equation 55.

$$d_j = \begin{cases} u_j & \text{if } 1 \le j \le \min(t,m) \\ c_{j-t} + u_j \,(\text{mod} p) & \text{otherwise} \end{cases} \tag{55}$$

2.8 Group 8: Key-Verifiable Multi Secret Sharing Schemes Type I

A fair amount of research has been done on (m,t,n) VMSSSs type I, half of which belong to this group. [99] extends from [94] by verifying whether keys shared between PTs are correct. In the sharing process, each key k_i, its signature s_k_i and public key v are created by Equations 56, 57 and 58, respectively, where prime p_1 is a multiple of prime p_2, $\{u_i\}_{i=0,\cdots,n}$ are random integers and ϕ is Euler's totient function [48]. Key k_i is stored at PT_i and $\{s\_k_i\}_{i=0,\cdots,n}$ and v are published on the NB. Before reconstruction, keys are verified. Key k_i is correct if $((s\_k_0)^{k_i})^v \equiv u_i' \mod p_1$.

$$k_i = (s\_k_i)^{u_0} \bmod p_1 \tag{56}$$
$$s\_k_i = (p_2)^{u_i} \bmod p_1 \tag{57}$$
$$v = (u_0)^{-1} \bmod \phi(p_1) \tag{58}$$

[27] also extends from [94] with the same goal. Only key and signature generation actually varies. However, the verification process is more efficient. Key k_i is created by Equations 59, 60 and 61, where $u_{i=1,2,3}$ are random integers and f is any two-variable one-way function. Signature s_k_i of key k_i is created by Equation 62, where u_4 is a random integer. Key k_i is stored at PT_i, while $u_1, \cdots, u_4$ and $\{s\_k_i\}_{i=1,\cdots,n}$ are published on the NB.

$$k_i = f(u_1, w_i) \tag{59}$$
$$w_i = ((v_i)^{u_3})^{u_2} \bmod p \tag{60}$$
$$u_2 \times u_3 \equiv 1 \bmod \phi(p) \tag{61}$$
$$s\_k_i = (u_4)^{k_i} \bmod p \tag{62}$$

[28] in turn extends from [27] by proposing new secret sharing and reconstruction processes to reduce computation costs. After keys and signatures are created, shares $\{c_{j,1}\}_{j=1,\cdots,n}$ and $\{c_{j,2}\}_{j=1,\cdots,m}$ are generated by Equations 63, 64, 65 and 66, where u_0 is a random integer. Next, $\{c_{j,1}\}_{j=1,\cdots,n}$ and $\{c_{j,2}\}_{j=1,\cdots,m}$ are published on the NB. After key verification, secrets are reconstructed by Equations 67, 68 and 69.

$$c_{j,1} = d_j - y_{j+n} \tag{63}$$
$$c_{j,2} = k_j - y_{j-1} \tag{64}$$
$$y_j = \begin{cases} k_{j+1} & \text{if } 0 \le j < t \\ -\sum_{v=1}^{t} u_v \times y_{j-v} \bmod p & \text{otherwise} \end{cases} \tag{65}$$
$$(x-u_0)^t = x^t + u_1 \times x^{t-1} + \cdots + u_t = 0 \tag{66}$$
$$d_j = y_{j+n} + c_{j,2} \tag{67}$$
$$y_j = \begin{cases} k_{j+1} & \text{if } 0 \le j < t \\ k_{j+1} - c_{j+1,1} & \text{if } t \le j < n \\ f(j) \times (u_0)^j \bmod p & \text{otherwise} \end{cases} \tag{68}$$
$$f(x) = \sum_{v=1}^{t} \frac{y_{v-1}}{(u_0)^{v-1}} \prod_{w=1 \& w \neq v}^{t} \frac{x-w+1}{v-w} \bmod p \tag{69}$$

[89] extends from [21] (Section 2.9) to improve the efficiency of the sharing and reconstruction processes. To this aim, secrets are split into blocks of size t that are each shared and reconstructed all at once. Block b_l is divided into n shares $\{c_{l,h}\}_{h=1,\cdots,n}$ by Equation 70, where $A = [a_{i,w}]_{t \times n}$, $a_{i,w} = H(u_l \times k_i \times v)^{w-1}$, and $\{u_l\}_{l=1,\cdots,o}$ and v are random integers. Key k_i is stored at PT_i, whereas key signatures $\{s\_k_i\}_{i=1,\cdots,n}$, shares $\{c_{l,h}\}_{l=1,\cdots,o;h=1,\cdots,n}$ and $\{x_l = u_l \times v\}_{l=1,\cdots,o}$ are published on the NB. To reconstruct secrets, shares and keys are verified for correctness with a bilinear map $f(u_l \times k_i \times v, v) = f(x_l, s\_k_i)$. Then, secrets are reconstructed by solving Equation 70.

$$[c_{l,1}, \cdots, c_{l,n}]^T = A \times [b_l]^T \tag{70}$$

[38] also extends from [21], pursuing the same goal as [89]. The difference is that secrets are divided into $n+m-t$ shares to reduce the number of shares. Shares $\{c_h\}_{h=1,\cdots,(n+m-t)}$ are computed by Equation 71, where $A = [a_{x,y}]_{(m+n)\times(m+n-t)}$, $a_{x,y} = (w)^{x(y-1)}$, $z_i = H(u \times v \times k_i)$, and u, v and w are random integers. Key k_i is stored at PT_i, whereas key signatures $\{s\_k_i\}_{i=1,\cdots,n}$, shares $\{c_h\}_{h=1,\cdots,(n+m-t)}$, data signatures $\{s\_d_j\}_{j=1,\cdots,m}$ and $x = u \times v$ are published on the NB. To reconstruct secrets, shares and keys are verified for correctness with a bilinear map $f(u \times k_i \times v, v) = f(x, s\_k_i)$. Then, secrets are reconstructed by solving Equation 71.

$$[c_1, \cdots, c_{n+m-t}]^T = A \times [z_1, \cdots, z_n, d_1, \cdots, d_m]^T \tag{71}$$

Unlike in other schemes, PTs in [20] can be added or deleted. Moreover, threshold t can vary. To this aim, keys k_i, key signatures s_k_i and PT identifiers ID_i are randomly selected such that they are different from one PT to the other. Then, secrets are organized into unfixed-sized blocks, where block b_l stores u_l secrets. All secrets $\{d_{l,q}\}_{q=1,\cdots,u_l}$ in block b_l are divided into $n+u_l-t_l$ shares $\{c_{l,h}\}_{h=1,\cdots,(n+u_l-t_l)}$ by Equations 72, 73, 74, 75 and 76, where z_l is a random integer. Each key k_i is stored at PT_i and identifiers $\{ID_i\}_{i=1,\cdots,n}$, signatures $\{s\_k_i\}_{i=1,\cdots,n}$ and shares $\{y_l\}_{l=1,\cdots,n}$ and $\{c_{l,h}\}_{l=1,\cdots,o; h=1,\cdots,(n+u_l-t_l)}$ are published on the NB. Before reconstruction, keys are verified for correctness with a discrete logarithm modulo and a one-way hash function. Finally, each secret $d_{l,q}$ in block b_l is reconstructed by Lagrange interpolation.

$$c_{l,h}=f_l\,(n+u_l+h) \tag{72}$$

$$f_l(x)=\sum_{v=1}^{u_l} d_{l,v}\times \Delta_1+\sum_{v=1}^{n}(s\_k_v)^{z_l}\times \Delta_2 \bmod p_1 \tag{73}$$

$$\Delta_1=\prod_{w=1\&w\neq v}^{u_l}\frac{x-(n+w)}{v-w}\times\prod_{i=1}^{n}\frac{x-ID_i}{(n+v)-ID_i} \tag{74}$$

$$\Delta_2=\prod_{i=1\&i\neq v}^{n}\frac{x-ID_i}{ID_v-ID_i}\times\prod_{w=1}^{u_l}\frac{x-(n+w)}{ID_v-(n+w)} \tag{75}$$

$$y_l=(p_2)^{z_l} \bmod p_1 \tag{76}$$

[60] extends from [72] to reduce computation cost and verify key correctness. Secrets are organized into blocks of size $t-1$. Keys $\{k_i\}_{i=1,\cdots,n}$ are randomly selected and their signatures $\{s\_k_i\}_{i=1,\cdots,n}$ are created by Equation 77, where H is a hash function. In block b_l, the first secret $d_{l,1}$ is divided into two shares $c_{l,1,1}$ and $c_{l,1,2}$ by Equation 78, where u is a random integer. Other secrets in block b_l are shared by Equations 79 and 80. Key k_i is stored at PT_i and $\{s\_k_i\}_{i=1,\cdots,n}$, $\{c_{l,q,h}\}_{l=1,\cdots,o;q=1,\cdots,t-2;\ h=1,\cdots,q+1}$ and $\{c_{l,t-1,h}\}_{l=1,\cdots,o; h=1,\cdots,n}$ are published on the NB. Before reconstruction, each key k_i is verified for validity by Equation 81. Then, all secrets in each block are reconstructed by Lagrange interpolation.

$$s\_k_i=H\left(H^{t-1}\left(k_i\right)\oplus k_i\right) \tag{77}$$

$$c_{l,1,h}=u\times h+d_{l,1}-\left(k_q\oplus H\left(k_i\right)\right) \tag{78}$$

$$c_{l,q,h}=f_{l,q}\left(h\right)-\left(k_q\oplus H^q\left(k_q\right)\right) \tag{79}$$

$$f_{l,q}(x)=\begin{cases} d_{l,q}+u\times x & \text{if } q=1\\ d_{l,q}+\sum_{v=1}^{q}x^v\times f_{l,q-1}\left(x\right) & \text{otherwise}\end{cases} \tag{80}$$

$$s\_k_i=H\left(H^{t-1}\left(k_i\right)\oplus k_i\right) \tag{81}$$

Finally, [52] propose two schemes. They create keys and verify their correctness by using a one-way hash function and a LFSR public key cryptosystem [40,41]. The first scheme shares and reconstructs secrets as [94], while the second scheme does as [28], while providing higher security than [94,28] with keys of same lengths.

2.9 Group 9: Key *and* Data-Verifiable Multi Secret Sharing Schemes Type I

The other third of (m,t,n) VMSSSs type I belong to this group. [82] prevents cheating from malicious PTs by verifying both shares and keys. Keys $\{k_i\}_{i=1,\cdots,n}$ and their signatures $\{s\_k_i\}_{i=1,\cdots,n}$ are created by Equations 82, 83, 84, 85 and 86, where $\{u_v\}_{v=0,\cdots,t-1}$ are random integers and $a_1,\cdots,a_5$ are set as discrete logarithms. Let p_1 and p_2 be big primes. a_1 is a random integer, $a_2=(2\times p_1+1)(2\times p_2+1)$, $a_3=p_1\times p_2$ and $a_3\times a_2=\phi(a_5)$, where ϕ is Euler's totient function [48]. Key k_i is stored at PT_i, while signatures $\{s\_k_i\}_{i=1,\cdots,n}$ and $\{w_v\}_{v=0,\cdots,t-1}$ are published on the NB. Key correctness is checked by Equation 87.

$$f(x)=\left(\sum_{v=0}^{t-1}u_v\times x^v\right) \bmod a_3 \tag{82}$$

$$w_v=(p_1)^{u_v} \bmod a_2 \tag{83}$$

$$y_i=\prod_{\forall PT_v,v\neq i}(ID_i-ID_v) \bmod a_3 \tag{84}$$

$$k_i=(f(ID_i)/y_i) \bmod a_3 \tag{85}$$

$$s\_k_i=(a_1)^{k_i} \bmod a_2 \tag{86}$$

$$((a_1)^{y_i})^{k_i}=\prod_{v=0}^{t-1}(w_v)^{(ID_i)^v} \bmod a_2 \tag{87}$$

A 4-tuple of shares $\{c_{j,1},\cdots,c_{j,4}\}$ is created by Equations 88 and 89, where $c_{j,1}$ and $c_{j,2}$ are random integers. Shares $\{c_{j,h}\}_{j=1,\cdots,m;h=1,\cdots,4}$ are published on the NB. Before reconstruction, each PT_i must verify share and key correctness by Equation 90. If verification is positive, secrets are reconstructed by Equations 91 and 92, where G is any group of t PTs.

$$c_{j,3}=(a_1)^{-a_5+c_{j,1}}\times(c_{j,2})^{2\times a_5+c_{j,1}+1} \bmod a_2 \tag{88}$$

$$c_{j,4}=((c_{j,2})^{u_0}-d_j)(c_{j,3})^{-u_0} \bmod a_2 \tag{89}$$

$$\begin{aligned}((c_{j,3})^{k_i})^{a_4}\equiv&(s\_k_i)^{a_4\times c_{j,1}-1}\times\\ &((c_{j,2})^{k_i})^{2+a_4(c_{j,1}+1)} \bmod a_2\end{aligned} \tag{90}$$

$$\begin{aligned}d_j=&\left(\textstyle\prod_{PT_i\in G}((c_{j,2})^{k_i})^{\triangle_i}\right)-\\ &\left(c_{j,4}\textstyle\prod_{PT_i\in G}((c_{j3})^{k_i})^{\triangle_i}\right) \bmod a_2\end{aligned} \tag{91}$$

$$\triangle_i=\prod_{\forall PT_v\in G}-ID_v\times\prod_{\forall PT_v\in G}(ID_i-ID_v) \tag{92}$$

[19] extends from [82] to improve the efficiency of the sharing and reconstruction processes. To this aim, j 3-tuples of shares $\{c_{j,1},c_{j,3},c_{j,4}\}_{j=1,\cdots,m}$ are created by Equations 93 and 94 and published on the NB. Before reconstruction, each PT must verify share and key correctness by Equation 95. If verification is positive, secrets are reconstructed by Equations 96 and 92.

$$c_{j,3}=(a_1)^{a_5\times c_{j,1}} \bmod a_2 \tag{93}$$

$$c_{j,4}=\left((a_1)^{u_0\times a_5\times c_{j,1}} \bmod a_2\right)\oplus d_j \tag{94}$$

$$((c_{j,3})^{k_i})^{a_4}\equiv(s\_k_i)^{c_{j,1}} \bmod a_2 \tag{95}$$

$$d_j=c_{j,4}\oplus\prod_{\forall PT_i\in G}((c_{j,3})^{k_i})^{\triangle_i} \bmod a_2 \tag{96}$$

[78] extends from [94] by checking whether keys and shares are valid, with the help of a discrete logarithm. Signatures $\{s\_d_j\}_{j=1,\cdots,\max(m,t)}$ are created after secrets are shared by Equation 97, where $\{u_j\}_{j=1,\cdots,m}$ are secrets ($u_j = d_j$) and $\{u_j\}_{j=(m+1),\cdots,t}$ are random integers. They are then published on the NB. Before reconstruction, keys are verified first, and then shares are, both by Equation 98. Signature s_d_j is also used to check share integrity.

$$s\_d_j=(p_1)^{u_j} \bmod p_2 \tag{97}$$

$$(p_1)^{c_i}=\prod_{h=1}^{\max(t,m)}(c_{h+n+1})^{f(w,k_i)^h} \bmod p_2 \tag{98}$$

In [21], each secret d_j is divided independently into vary threshold t_j. Keys $\{k_i\}_{i=1,\cdots,n}$ are randomly selected such that their signatures $\{s\_k_i\}_{i=1,\cdots,n}$ (Equation 99, where v is a random integer) are unique. Each secret d_j is divided into n shares $\{c_{j,h}\}_{h=1,\cdots,n}$ by Equations 100 and 101, where $A_j = [a_{x,y}]_{(n\times t_i)}$, $a_{x,y} = (u)^{x(y-1)}$, $Z_j = [w_j\times v, d_j\times(k_1)^v,\cdots,d_j\times(k_n)^v]$ and u and w_j are random integers. Signature s_d_j of d_j is created by Equation 102. Keys k_i are stored at PT_i, whereas key signatures $\{s\_k_i\}_{i=1,\cdots,n}$, shares $\{w_j, c_{j,1},\cdots,c_{j,n}\}_{j=1,\cdots,m}$, signatures $\{s\_d_j\}_{j=1,\cdots,m}$, u and v are published on the NB. Before reconstruction, shares and keys are verified for correctness with a bilinear map $f((k_i)^{s\_d_j}, v) = f(s\_d_j,(k_j)^v)$. Then, secrets are reconstructed by solving linear Equations 100 and 101.

$$s\_k_i=(k_i)^v \tag{99}$$

$$[c_{j,1},\cdots,c_{j,n}]^T=A_j\times[Z_j]^T \tag{100}$$

$$d_j=H(w_j\times v) \tag{101}$$

$$s\_d_j=d_j\times v \tag{102}$$

Unlike other schemes that compute integers over a finite field, [25] exploits binary strings in all processes to improve the efficiency of both sharing and reconstruction processes. In the sharing process, two kinds of keys are randomly created in binary string format: PT keys $\{k_i\}_{i=1,\cdots,n}$ and user keys $\{u_{j,v}\}_{j=1,\cdots,m;v=1,\cdots,t_l}$. Then, each share $c_{j,h}$ is created by Equation 103, where H is a one-way hash function and $\|$ is the concatenation operator. Finally, shares $c_{j,h}$, $H(d_j)$, $H(H(k_i\parallel j\parallel h))$ with $j = 1,\cdots,m$; $h = 1,\cdots,t_l$ and $i = 1,\cdots,n$, are published on the NB.

$$c_{j,h}=d_j\oplus\left\{\oplus_{i:PT_i\in u_{j,v}}H(k_i\parallel j\parallel h)\right\} \tag{103}$$

Secrets are reconstructed by Equation 104 if all keys pass the verification process, which is split in two steps. Before reconstruction, keys $\{k_i\}_{i=1,\cdots,n}$ are checked for correctness by comparison with signatures $H(H(k_i\parallel j\parallel h))$. After reconstruction, secrets $\{d_j\}_{j=1,\cdots,m}$ are checked for correctness by comparison with signatures $H(d_j)$.

$$d_j=c_{l,h}\oplus\left\{\oplus_{i:PT_i\in u_{j,v}}H(k_i\parallel j\parallel h)\right\} \tag{104}$$

Finally, [16] extends from [77] by sharing multiple secrets, to improve sharing/reconstruction efficiency and reduce share volume. To this aim, PT_i's identifier ID_i is randomly selected and PT_i's key k_i and signatures $\{s\_k_v\}_{v=0,\cdots,(t-1)}$ are created by Equations 105 and 106, respectively, where x and y are randomly created with NTRU [74] and w is NTRU's blinding value. Each secret d_j is divided into a 3-tuple of shares $\{c_{j,1}, c_{j,2}, c_{j,3}\}$ by Equations 107 and 108, where $c_{j,1}$ is a random integer. Key k_i is stored at PT_i, whereas identifiers $\{ID_i\}_{i=1,\cdots,n}$, signature $\{s\_k_v\}_{v=0,\cdots,(t-1)}$ and shares $\{c_{j,h}\}_{j=1,\cdots,m;h=1,\cdots,3}$ are published on the NB.

$$k_i=\sum_{v=0}^{t-1}u_v\times(ID_i)^v \tag{105}$$

$$s\_k_v=w\times x+u_v \bmod p_1 \tag{106}$$

$$c_{j,2}=w\times x+c_{j,1} \bmod p_1 \tag{107}$$

$$c_{j,3}=d_j\oplus H(u_0\times c_{j,2}) \tag{108}$$

Before reconstruction, keys and shares are verified for correctness by Equations 109 and 110, respectively. Finally, secrets are reconstructed by Equation 111.

$$k_i=y\sum_{v=0}^{t-1}s\_k_v(ID_i)^v \bmod p_2 \tag{109}$$

$$y\times k_i\times c_{j,2}=y\sum_{v=0}^{t-1}(w_v\times(ID_i)^v\times c_{j,1}) \bmod p_2 \tag{110}$$

$$d_j=c_{j,3}\oplus H\left(\sum_{i\in G}k_i\times c_{j,2}\times\prod_{v\in G\&v\neq i}\frac{-ID_v}{ID_i-ID_v}\right) \tag{111}$$

2.10 Group 10: Data-Verifiable Multi Secret Sharing Schemes Type II

VMSSSs type II are recent. Unlike all previous SSSs, [6] verifies both PT honesty and share correctness with inner and outer signatures, respectively. Inner signatures are signatures that help verify secret correctness after reconstruction. If one or more shares are erroneous, then reconstructed secrets do not match with their inner signatures. Outer signatures are share signatures. The correctness of shares is checked before reconstructing secrets.

In the sharing process of [6], n distinct random linear equations $\{f_i\}_{i=1,\cdots,n}$ (Equation 112, where coefficients $u_{i,v}$ are random positive integers) are created. Then, m secrets $\{d_{l,q}\}_{q=1,\cdots,t-1}$ are organized into o blocks b_l of size $t-1$. The inner signature s_b_l of block b_l is created with the help of an homomorphic function (Equation 113). Next, n shares $\{e_{l,i}\}_{i=1,\cdots,n}$ are created by Equation 114. Their outer signatures $\{s\_out_{l,i}\}_{i=1,\cdots,n}$ are created with any hash function. Shares $\{e_{l,i}, s\_out_{l,i}\}$ $_{l=1,\cdots,o}$ are stored at PT_i.

$$f_i(x_1,\cdots,x_t)=x_t \times u_{i,v} + \sum_{v=1}^{t-1}(x_v+2)\times u_{i,v} \tag{112}$$

$$s\_b_l=H(b_l) \tag{113}$$

$$e_{l,i}=f_i(b_l, s\_b_l) \tag{114}$$

Before reconstruction, shares from t out of n PTs are verified against their outer signatures. Then, blocks and their inner signatures are reconstructed by solving the linear equations. Finally, recovered blocks are verified against their inner signatures. If the test fails, erroneous blocks can be reconstructed from shares in a new PT group.

[5] extends from [76] by sharing each secret at fewer than n PTs'. PT failure is also allowed, more specifically by allowing data updates at remaining online PTs. Moreover, [5] also protects from PT *group* cheating by imposing a new constraint: no PT group can hold enough shares to reconstruct the secret when $n < 2t-2$. PT honesty and share correctness are checked as in [6]. In addition, this scheme separates outer signature creation and verification from the sharing and reconstruction processes.

Although [5] is an MSSS, each secret is shared and reconstructed independently. Inner signature s_d_j of secret d_j is created with the help of an homomorphic function. Next, PTs are split into two groups: $n-t+2$ PTs in group G_1 and $t+2$ PTs in group G_2. Then, $t+2$ pseudo shares $\{e_{j,i}\}_{PT_i\in G_2}$ (G_2's shares created to construct polynomial f_2 but not stored at $PT_i \in G_2$) are created from d_j's identifier d_id_j and identifiers $\{ID_i\}_{PT_i\in G_2}$ of PTs in G_2 with an homomorphic function (Equation 115).

$$e_{j,i}=f_1(d\_id_j, ID_i) \tag{115}$$

Next, a polynomial f_2 of degree $t-1$ is created from d_j, inner signature s_d_j, pseudo shares $\{e_{j,i}\}_{PT_i\in G_2}$ and PT identifiers $\{ID_i\}_{PT_i\in G_2}$ by Lagrange interpolation (Equation 116, where $\{(x_1,y_2),\ldots,(x_t,y_t)\} = \{(H(K_d),d_j),(H(K_s),s\_d_j)\}\cup\{(H(ID_i),e_{j,i})_{PT_i\in G_2}\}$).

$$f_2(x)=\sum_{u=1}^{t}\prod_{1\le v\le t, u\ne v}\frac{x-x_v}{x_u-x_v}\times y_u \tag{116}$$

Shares $\{e_{j,i}\}_{PT_i\in G_1}$ are created by Equation 117 and stored at $PT_i \in G_1$. To reconstruct d_j, t out of n PTs from G_1 and G_2 are selected. Secrets are reconstructed by Lagrange interpolation (Equation 116) from both shares and pseudo shares (Equation 115).

$$e_{j,i}=f_2(H(IDi)) \tag{117}$$

2.11 Group 11: Key *and* Data-Verifiable Multi Secret Sharing Schemes Type II

[80] is the only (m,t,n) VMSSS type II. It exploits elliptic curve cryptography to verify the correctness of both shares and keys. In the sharing process, keys $K = \{k_{i,q}\}_{i=1,\cdots,n,q=1,\cdots,t}$ are randomly chosen from small integers. Then, $l\times t$ secrets $\{d_{l,q}\}_{l=1,\cdots,o;q=1,\cdots,t}$ are organized into o blocks $\{b_l\}_{l=1,\cdots,o}$ of size t. Each block b_l is divided into n shares $\{e_{l,i}\}_{i=1,\cdots,n}$ by Equation 118. Signature $s\_d_{l,q}$ of $d_{l,q}$ is created by Equation 119, where u is an elliptic curve point. Keys $\{k_{i,q}\}_{q=1,\cdots,t}$ and shares $\{e_{l,i}\}_{l=1,\cdots,o}$ are stored at PT_i, whereas signatures $\{s\_d_{l,q}\}_{l=1,\cdots,o;q=1,\cdots,t}$ are published on the NB. Before reconstruction, each share $e_{l,i}$ and its keys $\{k_{i,q}\}_{q=1,\cdots,t}$ are verified for correctness by Equation 120. Finally, each block is reconstructed by solving t simultaneous linear equations (Equation 118).

$$[e_{l,1},\cdots,e_{l,n}]^T=K\times[b_l]^T \bmod p \tag{118}$$

$$s\_d_{l,q}=u\times d_{l,q} \tag{119}$$

$$u\times[e_{l,1},\cdots,e_{l,n}]^T=K\times[s\_d_{l,1},\cdots,s\_d_{l,t}]^T \tag{120}$$

3 Discussion

In this section, we compare the SSSs presented in Section 2 along four axes. First, we provide a global view of the evolution of SSSs since their inception (Section 3.1). Second, we synthesize and account for the various sharing and verification techniques used in SSSs to enforce data security (Section 3.2). Third, we compare the features provided by SSSs beyond data privacy and integrity (Section 3.3). Finally, we study the factors that influence the cost of cloud SSS-based solutions in the pay-as-you-go paradigm (Section 3.4).

3.1 Evolution of Secret Sharing Schemes

To clarify the historical relationships between the SSSs reviewed in this paper and better visualize the improvements brought to Shamir's [76] and Blakley's [12] schemes since 1979, we refer the reader to Figure 10. In this flowchart, each scheme is identified by a bibliographical reference (in red), the group (in orange) and type (in yellow) it belongs to (Section 2), and whether it enforces key (represented by a green K) and/or data (represented by a blue D) verification. Moreover, a brief text describes the novelty brought by each scheme. Finally, an arrow from scheme S_1 to scheme S_2 indicates that S_1 extends from S_2. For example, [78], proposed in 2005, is a VMSSS type I belonging to Group 9. This scheme can verify both data and key correctness and extends from [94] to improve sharing and reconstruction efficiency.

Figure 10 quite clearly shows that SSSs have been less studied for almost 25 years than since the 2000's, when they attracted new attention in conjunction with the development of new, on-line distributed systems, i.e., clusters, grids and the cloud. Moreover, research about secret sharing seems to have accelerated since 2012, with the wide spread of cloud computing and associated data security concerns.

3.2 Sharing, Reconstruction and Verification Methods

SSSs may be subdivided into five subprocesses, i.e., data sharing, data reconstruction, key creation, key verification and data verification. Of course, data sharing and reconstruction are the main processes for all groups of SSSs (Table 1). Key creation is always optional. Finally, data verification is the focus of groups 4, 6, 7, 9, 10 and 11; and key verification the focus of groups 5, 6, 8, 9 and 11. The methods supporting these processes in each studied SSS are summarized in Table 3.

Approximately half of the surveyed SSSs share secrets by polynomial interpolation and reconstruct them by Lagrange interpolation, as Shamir's [76]. Yet, other methods, such as homomorphic encryption, NTRU or RSA enhance security. Similarly, approximately half of the schemes necessitating keys generate them at random,while more elaborate methods such as hash functions, LFSR, NTRU or RSA help protect keys. Eventually, the same variety of methods is found in the key and data verification processes, although discrete logarithm modulo and hash functions are by far the most popular.

Given such variety, it is difficult to crisply rank the security level of all studied schemes. SSSs have indeed been continually addressing different issues over time, and thus adopted ad-hoc methods suited to their objectives. Moreover, the papers describing them typically do not compare to one another. Thence, we push the comparison of SSSs' features and cost in the following subsections.

3.3 Features of Secret Sharing Schemes

SSSs mainly aim at enforcing data security (privacy, availability and integrity). However, in the context of cloud data processing, efficient data access (update, search and aggregation operations) must also be made possible by SSSs. Thus, some SSSs allow computation (e.g., sums and averages [5,6,12,49,76,80] and exact matches [6,12,25]) directly over shares, i.e., without reconstructing secrets. To provide a global overview, the features of all studied SSSs are synthesized in Table 4, where an X means a particular feature is supported by the corresponding SSS(s); NB means that data availability is supported, but only when the NB is accessible; G means that data availability is supported only when shares are replicated; IN and OUT stand for inner and outer code verification, respectively; B means that updates operate on data blocks instead of individual shares; and I means that exact matches are run on indices.

3.3.1 Data Privacy and Availability

Since all SSSs divide data into n shares such that each individual share is meaningless, they enforce data privacy by design. Moreover, data availability is guaranteed as long as t out of n PTs are available, since t PTs are enough to reconstruct secrets. However, a coalition of t or more malicious PTs can break any secret. Thus, [5] provides further privacy by protecting data from PT group cheating, by having a number of shares at all PTs that is lower than t. Finally, since most (V)MSSSs type-I store all shares in the NB, they are vulnerable and can loose data access if the NB is compromised.

The privacy level of all SSSs mainly depends on parameter t. Provided PTs independently enforce sound

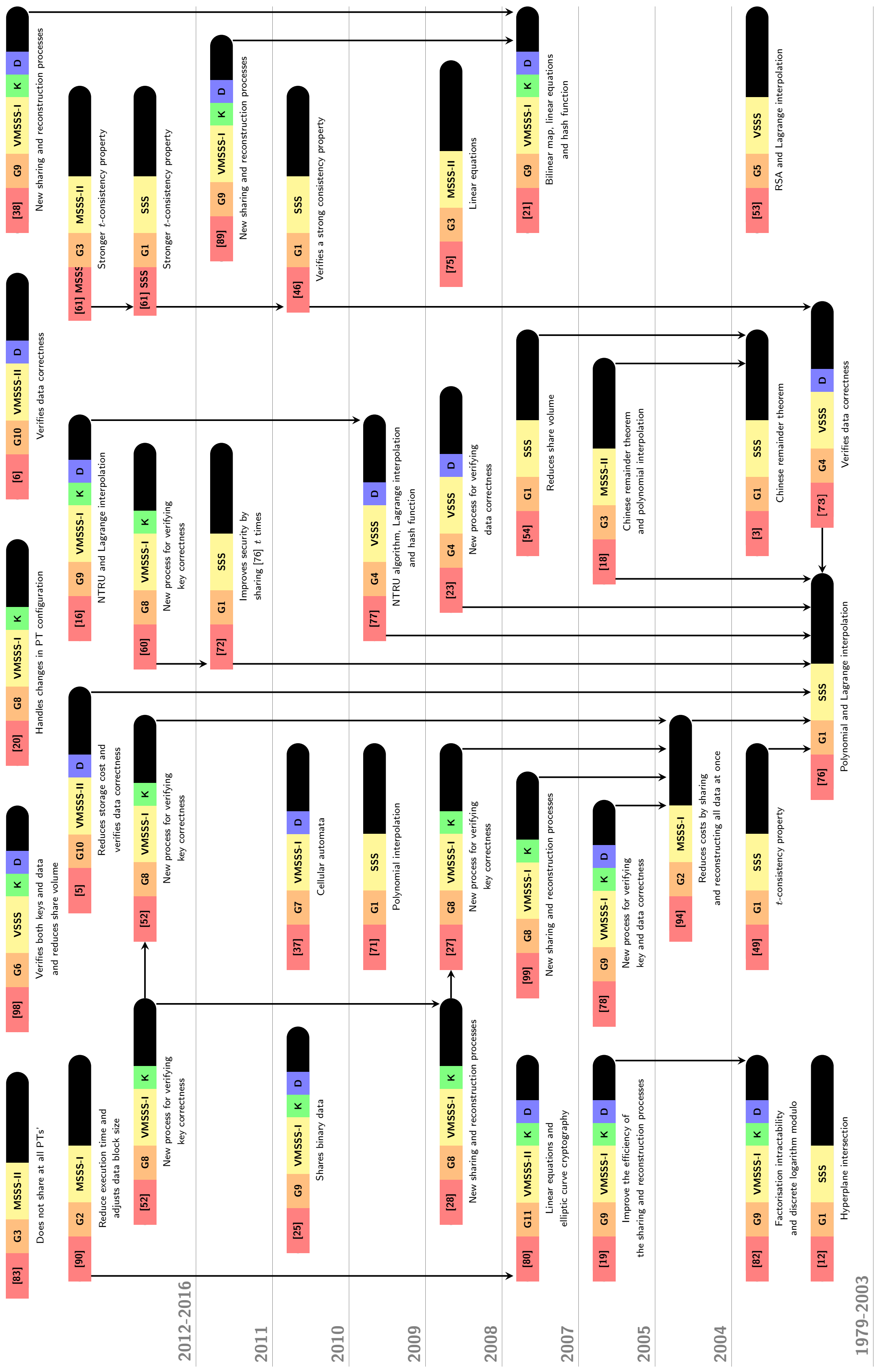

2012-2016
2011
2010
2009
2008
2007
2005
2004
1979-2003
[83] G3 MSSS-II
Does not share at all PTs'
[98] G6 VSSS K D
Verifies both keys and data and reduces share volume
[20] G8 VMSSS-I K
Handles changes in PT configuration
[6] G10 VMSSS-II D
Verifies data correctness
[38] G9 VMSSS-I K D
New sharing and reconstruction processes
[90] G2 MSSS-I
Reduce execution time and adjusts data block size
[5] G10 VMSSS-II D
Reduces storage cost and verifies data correctness
[16] G9 VMSSS-I K D
NTRU and Lagrange interpolation
[61] MSSS G3 MSSS-II
Stronger t-consistency property
[52] G8 VMSSS-I K
New process for verifying key correctness
[52] G8 VMSSS-I K
New process for verifying key correctness
[60] G8 VMSSS-I K
New process for verifying key correctness
[61] SSS G1 SSS
Stronger t-consistency property
[72] G1 SSS
Improves security by sharing [76] t times
[89] G9 VMSSS-I K D
New sharing and reconstruction processes
[25] G9 VMSSS-I K D
Shares binary data
[37] G7 VMSSS-I D
Cellular automata
[46] G1 SSS
Verifies a strong consistency property
[71] G1 SSS
Polynomial interpolation
[77] G4 VSSS D
NTRU algorithm, Lagrange interpolation and hash function
[28] G8 VMSSS-I K
New sharing and reconstruction processes
[27] G8 VMSSS-I K
New process for verifying key correctness
[23] G4 VSSS D
New process for verifying data correctness
[75] G3 MSSS-II
Linear equations
[80] G11 VMSSS-II K D
Linear equations and elliptic curve cryptography
[99] G8 VMSSS-I K
New sharing and reconstruction processes
[54] G1 SSS
Reduces share volume
[21] G9 VMSSS-I K D
Bilinear map, linear equations and hash function
[19] G9 VMSSS-I K D
Improve the efficiency of the sharing and reconstruction processes
[78] G9 VMSSS-I K D
New process for verifying key and data correctness
[18] G3 MSSS-II
Chinese remainder theorem and polynomial interpolation
[94] G2 MSSS-I
Reduces costs by sharing and reconstructing all data at once
[82] G9 VMSSS-I K D
Factorisation intractability and discrete logarithm modulo
[49] G1 SSS
t-consistency property
[3] G1 SSS
Chinese remainder theorem
[53] G5 VSSS
RSA and Lagrange interpolation
[12] G1 SSS
Hyperplane intersection
[76] G1 SSS
Polynomial and Lagrange interpolation
[73] G4 VSSS D
Verifies data correctness

Fig. 10: Evolution of SSSs

Table 3: Sharing, Reconstruction and Verification Methods in SSSs

Type	Group	Scheme	Method				
			Data sharing	**Data reconstruction**	**Key creation**	**Key verification**	**Data verification**
SSS	Group 1	[76]	Polynomial interpolation	Lagrange interpolation			
		[12]	Hyperplane intersection				
		[3, 54]	Chinese remainder theorem		Random		
		[49]	Polynomial interpolation	Lagrange interpolation	Random		
		[71]	Polynomial interpolation		Random		
		[46],[61] SSS	Homomorphism and polynomial interpolation	Lagrange interpolation			
		[72]	Polynomial interpolation and recursion				
MSSS	Group 2	[94]	Polynomial interpolation	Lagrange interpolation	Two-variable one-way function		
		[90]	Linear equation and matrix multiplication		Random		
	Group 3	[18]	Chinese remainder theorem and polynomial interpolation		Random		
		[75]	Linear equation and matrix multiplication		Random		
		[61] MSSS	Polynomial interpolation and homomorphism	Lagrange interpolation			
		[83]	Polynomial interpolation and pseudo-random number generation	Lagrange interpolation	Random		
VSSS	Group 4	[73]	Homomorphism and polynomial interpolation	Lagrange interpolation			RSA cryptosystem
		[23]	Polynomial interpolation	SMC			Discrete logarithm modulo
		[77]	NTRU and one-way hash function	Lagrange interpolation	NTRU		NTRU
	Group 5	[53]	RSA cryptosystem and Lagrange interpolation		RSA cryptosystem		
	Group 6	[98]	Jordan matrix and linear equations		Random	Discrete logarithm modulo	Discrete logarithm modulo
VMSSS	Group 7	[37]	One-dimensional cellular automaton				Discrete logarithm modulo
	Group 8	[99]	Polynomial interpolation	Lagrange interpolation	Discrete logarithm modulo		
		[27]	Polynomial interpolation	Lagrange interpolation	Discrete logarithm modulo and two-variables one-way function	Discrete logarithm modulo	
		[28]	Homogeneous linear recursion	Lagrange interpolation	Discrete logarithm modulo and two-variables one-way function	Discrete logarithm modulo	
		[89, 38]	linear equations and hash function		Random	Bilinear map	
		[20]	Discrete logarithm modulo and Lagrange interpolation		Random	Discrete logarithm modulo and one-way hash function	
		[60]	Polynomial interpolation, recursion, XOR and one-way hash function	Lagrange interpolation, recursion, XOR and one-way hash function	XOR and one-way hash function		
		[52]-I	Polynomial interpolation	Lagrange interpolation	One-way hash function and LFSR public key cryptography		
		[52]-II	Homogeneous linear recursion	Lagrange interpolation	One-way hash function and LFSR public key cryptography		
	Group 9	[82, 19]	Factorisation intractability and discrete logarithm modulo		Discrete logarithm modulo	Factorisation intractability and discrete logarithm modulo	
		[78]	Polynomial interpolation	Lagrange interpolation	Two-variable one-way function	Discrete logarithm modulo	
		[21]	linear equations and hash function		Random	Bilinear map	
		[25]	One-way hash function and binary operations		Random binary	One-way hash function	
		[16]	NTRU, XOR and one-way hash function		NTRU	NTRU and one-way hash function	
	Group 10	[6]	Linear equations and homomorphic function		Random	One-way hash function	
		[5]	Lagrange interpolation, homomorphic function and hash function		Random	Homomorphic functions and tree structure	
	Group 11	[80]	Linear equations		Random	Elliptic curve cryptography	

Table 4: Features of SSSs

Type	Group	Scheme(s)	Features											
			Data privacy	Data availability	Key availability	Data integrity	Key integrity	Data access						Other features
								Updates				Exact match	Aggregation	
								Add	Delete	Append	Modify			
SSS	Group 1	[76]	X	X				X	X	X			SUM, AVG	
		[12]	X	X				X	X	X		X	SUM, AVG	
		[3,54,71,72]	X	X				X	X	X				
		[49]	X	NB				X	X	X			SUM, AVG	Strong t-consistency property
		[46,61]SSS	X	X				X	X	X				Strong t-consistency property
MSSS	Group 2	[94]	X	NB	X									
		[90]	X		X			B	B					t and block size may vary
	Group 3	[18]	X	X	X									t may vary
		[75]	X	NB	NB			B	B					t may vary
		[61]MSSS	X	X				X	X	X				Strong t-consistency property
		[83]	X	X				X	X	X				
VSSS	Group 4	[73,77]	X	X		OUT		X	X	X				
		[23]	X	X		IN		X	X	X				
	Group 5	[53]	X	G	X		OUT	X	X	X				n may vary
	Group 6	[98]	X	NB	X	OUT	OUT	X	X	X				
VMSSS	Group 7	[37]	X	NB		OUT								
	Group 8	[99,27,28,52,38]	X	NB	X		OUT							
		[20]	X	NB	X		OUT	B	B					n and t may vary
		[89,60]	X	NB	X		OUT	B	B					
	Group 9	[82,19,16]	X	NB	X	OUT	OUT	X	X	X				
		[78]	X	NB	X	OUT	OUT							
		[21]	X	NB	X	OUT	OUT	X	X	X				t may vary
		[25]	X			IN	OUT	X	X	X		X		
	Group 10	[6]	X	X		INT, OUT		B	B			X	SUM, AVG	
		[5]	X	X		IN, OUT		X	X	X		I	SUM, AVG	Can share new data although some PTs disappear
	Group 11	[80]	X	X	X	OUT	OUT	B	B				SUM, AVG	

security measures, collecting at least t shares, i.e., compromising at least t PTs, is indeed harder and harder when t increases. High data protection is thus achieved when t is large [3,26], but at the expense of computing overhead, especially when sharing and reconstructing data (Section 3.4). Moreover, some SSSs may be insecure for applications where t is limited in practice. For instance, when t is a number of CSPs or servers, budget constraints come into play. We discuss three frameworks for outsourcing data in the cloud that address this issue in Section 4.

The robustness of almost SSSs directly relies on the gap between the two parameters n and t. The secret can be recovered although up to $n - t$ PTs disappear. Nevertheless, computing time and storage costs become prohibitive when $n \gg t$ (Section 3.4). Thus, n should be only a little bigger than t to achieve data availability with acceptable costs.

3.3.2 Data Integrity

The reconstruction process in SSSs always produces the correct result if secrets, shares and sharing and reconstruction functions are defined over a finite field [8]. However, if shares are altered, reconstructed secrets are mechanically incorrect. Thus, VSSSs and VMSSSs have been introduced to enforce data integrity. We categorize them into four classes: SSSs that verify keys, shares, secrets or both secrets and shares.

First, all schemes in groups 5, 6, 8, 9 and 11 verify keys before reconstructing shares. Hence, they can detect PT cheating and prevent transferring any data back to the user when incorrect keys are detected.

Second, most schemes in groups 4, 6, 7, 9, 10 and 11 verify the correctness of shares before reconstruction to reduce computation cost at the user's (no reconstruction occurs from incorrect shares). However, they require extra storage for signatures.

Third, [23,25] verify the correctness of reconstructed secrets. Their signature volumes are lower than that of the second class of VSSSs, since the number of shares is generally greater than that of secrets. However, incorrect secrets are detected only after they are already reconstructed.

Fourth, [6,5] verify the correctness of both secrets and shares with inner and outer code verification, respectively. Thus, no erroneous share is transferred to the user. Moreover, any PT cheating is detected.

Finally, although VSSSs and MVSSSs guarantee integrity, they consume more storage to handle signatures and more CPU power to verify keys, shares, and/or secrets. Moreover, to achieve the best possible verification performance, i.e., the lowest possible false positive rate, signatures must be big [5,6]. A larger storage volume is thus required. We push the comparison of such costs in Section 3.4.

3.3.3 Data Access

SSSs manage data at two levels: data piece or data block. First, [83,5] and most schemes in groups 1, 4, 5, 6, 9 share secrets independently. Hence, they can directly update data. For example, any secret can be deleted by removing its shares at all PTs'. Second, [90, 75,89,20,60,80,6] share secrets as blocks and support the homomorphic property. Thus, they allow updating shared blocks without reconstruction. Moreover, they update data faster because several shares in the same data block can be updated at once. In contrast, the schemes that share all secrets at once cannot perform updates on shares. The whole database must indeed be reconstructed, updated and then shared again. Thus, such schemes require longer execution times and use lots of memory when updating data.

Some SSSs allow computing exact matches on shares. Since [76,12,49,80,6,5] use polynomial or linear equations to share data, they also allow sum and average operations on shares. Moreover, [12,25,6] allow exact matches on shares, because they use the same keys to share all secrets. In contrast, [5] uses indices to achieve exact match queries. Indices indeed help perform faster exact matches than operating directly on shares, although at the expense of extra storage volume. Thus, the tradeoff between security and query efficiency must be carefully considered before choosing an SSS. We further discuss this issue in Sections 4 and 5.2.

3.3.4 Other Features

More features are included in some schemes. [49,46,61] verify a strong t-consistency property. Thus, they guarantee that any subset of t shares or more always reconstruct the same data, but that any subset of t shares or fewer cannot. [53,20] allow the user to add and remove PTs to/from the PT pool by updating the value of n. [90,18,75,20,21] allow the user to assign different values of t to different secrets, to enforce different security levels for each secret. Eventually, [5] allows inserting new data even if some PTs disappear.

3.4 Costs

In the cloud pay-as-you-go paradigm, the cost of securing data must be balanced with the risk of data loss or pilfering, and thus the level of data security must be balanced with its cost. This is a particularly important issue with secret sharing, which basically multiplies secret data volume by n in the worst case (provided individual share volume is not greater than secret data volume). We summarize the costs induced by SSSs in Table 5.

SSS time complexity and storage volume depend on a few parameters: m, n and t. To determine time complexity and storage volume, we suppose that only m is big. Other parameters n and t should remain quite small, because they relate to the number of PTs, i.e., the number of cloud service providers, which is limited in practice. Moreover, some SSSs such as [54,72,98] cannot assign a big value to parameters n and t because neither can be greater than the size of a secret.

3.4.1 Time Complexity

Data sharing and reconstruction complexity of most SSSs increases with n and t. In practice, n is a little bigger than t to guarantee data availability. Thus, the time complexity of sharing data is a little higher than that of reconstruction, e.g., $O(mnt) > O(mt^2)$ in [76]. However, when availability is not enforced, data sharing and reconstruction complexity is the same.

In contrast, in most MSSSs type I, secret sharing time complexity is clearly lower than that of data reconstruction, e.g., $O\left((n+m-t)\,t\right) < O(m^3)$ in [94], because they share several secrets at once but reconstruct each secret independently.

Overall, time complexities to share/reconstruct data by [37] are the lowest: $O(\max(m,t^2))$. Execution time actually depends only on m, because m is large, while both t and n are small in the normal case ($m \gg n \geq t$).

Moreover, VSSSs and VMSSSs must verify the correctness of keys and/or data. Thus, extra computation time is required. The time complexity of data/key verification is generally lower than that of data sharing/reconstruction. Moreover, the time complexity of key verification is generally lower than that of data verification. Several schemes achieve the lowest key verification complexity: $O(t)$, but only [37] achieves the lowest data verification complexity: $O(n)$.

3.4.2 Storage Volume

Figure 11 plots the estimated global share volume of all SSSs with respect to n, with $t = n - 1$ and original data volume is 1 GB. [60] is not plotted because global share volume grows very rapidly (about 252 GB when $n = 7$).

Almost all SSSs require a volume about n times that of secret data to store shares. Some SSSs propose solutions to minimize share volume. We categorize them

Table 5: Costs induced by SSSs

Type	Group	Scheme(s)	Average time complexity				Storage volume						
			Sharing process	Reconstruction process	Key verification	Data verification	Shares		Keys			Signatures	
							PTs	NB	PTs	NB	Client	PTs	NB
SSS	Group 1	[76]	$O(mnt)$	$O(mt^2)$			$mn\lVert d\rVert$						
		[12]	$O(mnt)$	$O(mt^2)$			$mn\lVert d\rVert$				$mt^2\lVert k\rVert$		
		[3]	$O(mn)$	$O(mt\beta)$			$mn\lVert d\rVert$				$(n+1)\lVert k\rVert$		
		[49]	$O(mnt)$	$O(mt^2)$				$mn\lVert d\rVert$	$2mn\lVert k\rVert$				
		[54]	$O(mn)$	$O(mt\beta)$			$\frac{mn\lVert d\rVert}{t}$				$(n+1)\lVert k\rVert$		
		[71]	$O(mn^2t)$	$O(mt^3)$			$mn\lVert d\rVert$		$nt\lVert k\rVert$				
		[46,61]SSS	$O(mn^2t)$	$O(mt^3)$			$mn\lVert d\rVert$						
		[72]	$O(mn^2t)$	$O(mt^3)$			$mn\lVert d\rVert/t$						
MSSS	Group 2	[94]	$O(nt)$ if $m \le t$ $O((n+m-t)t)$ otherwise	$O(\max(m^3, mt^2))$				$n\lVert d\rVert$ if $m \le t$ $(n+m-t)\lVert d\rVert$ otherwise	$n\lVert k\rVert$				
		[90]	$O(\max(nt\beta, mt^2))$	$O(\max(t^3, mt^2))$				$m\lVert d\rVert$	$n\lVert k\rVert$	$nt\lVert k\rVert$			
	Group 3	[18]	$O(mnt)$	$O(\max(mt^2, mt\beta))$			$mn\lVert d\rVert$		$n\lVert d\rVert$		$2m\lVert k\rVert$		
		[75]	$O(mn)$ or $O(bnt)$	$O(mt^2)$ or $O(bnt^3)$			$\ge mn\lVert d\rVert/t$ or $bn\lVert d\rVert$			$\approx nt\lVert k\rVert$			
		[61]MSSS	$O(mnt)$	$O(mt^2)$			$mn\lVert d\rVert$						
		[83]	$O(mnt)$	$O(mt^2)$			$(mn+m\gamma+\gamma)\lVert d\rVert$		$\beta\lVert k\rVert$				
VSSS	Group 4	[73]	$O(mn^2t)$	$O(mt^3)$		$O(mt^2)$	$mn\lVert d\rVert$						$mt\lVert s\rVert$
		[23]	$O(mnt)$	$O(mt^2)$		$O(m)$	$mn\lVert d\rVert$				$4\lVert k\rVert$		$m\lVert s\rVert$
		[77]	$O(mnt)$	$O(mt^2)$		$O(mt)$	$mn\lVert d\rVert$		$2n\lVert k\rVert$	$n\lVert k\rVert$	$4\lVert k\rVert$		$mn\lVert s\rVert$
	Group 5	[53]	$O(mt^2g)$	$O(mt^2)$	$O(t)$			$gm\lVert d\rVert$	$n\lVert k\rVert$	$n\lVert k\rVert$			$n\lVert s\rVert$
	Group 6	[98]	$O(\max(\gamma t, mt^2))$	$O(\max(\gamma t, mt^2))$	$O(t)$	$O(t^2)$	$\gamma m\lVert d\rVert/t^2$	$mn\gamma\lVert d\rVert/t^2$	$n\lVert k\rVert$	$(2\gamma t+2)\lVert k\rVert$			$(mn\gamma+n+1)\lVert s\rVert$
VMSSS	Group 7	[37]	$O(\max(m, t^2))$	$O(\max(m, t^2))$		$O(n)$	$n\lVert d\rVert$	$(m-t)\lVert d\rVert$ if $t > m$					$n\lVert s\rVert$
	Group 8	[99]	$O(nt)$ if $m \le t$ $O((n+m-t)t)$ otherwise	$O(\max(m^3, mt^2))$	$O(t)$			$n\lVert d\rVert$ if $m \le t$ $(n+m-t)\lVert d\rVert$ otherwise	$n\lVert k\rVert$	$\lVert k\rVert$	$(n+2)\lVert k\rVert$		$(n+1)\lVert s\rVert$
		[27]	$O(nt)$ if $m \le t$ $O((n+m-t)t)$ otherwise	$O(\max(m^3, mt^2))$	$O(t)$			$n\lVert d\rVert$ if $m \le t$ $(n+m-t)\lVert d\rVert$ otherwise	$n\lVert k\rVert$	$4\lVert k\rVert$			$n\lVert s\rVert$
		[28]	$O(m+n)$	$O(\max(m, t^2))$	$O(t)$			$(m+n)\lVert d\rVert$	$n\lVert k\rVert$	$4\lVert k\rVert$			$n\lVert s\rVert$
		[89]	$O(mnt)$	$O(mt^2)$		$O(mt)$		$mn\lVert d\rVert/t$	$n\lVert k\rVert$				$n\lVert s\rVert$
		[38]	$O(\max(m^2, t^2))$	$O(\max(m^2, t^2))$		$O(mt)$		$(n+m+t)\lVert d\rVert$	$n\lVert k\rVert$				$(n+m)\lVert s\rVert$
		[20]	$O(mnt^2)$	$O(mnt^2)$	$O(t)$			$mn\lVert d\rVert/t$	$n\lVert k\rVert$	$2n\lVert k\rVert$			$n\lVert s\rVert$
		[52]-I	$O(nt)$ if $m \le t$ $O((n+m-t)t)$ otherwise	$O(\max(m^3, mt^2))$	$O(t)$			$n\lVert d\rVert$ if $m \le t$ $(n+m-t)\lVert d\rVert$ otherwise	$n\lVert k\rVert$				$2n\lVert s\rVert$
		[52]-II	$O(m+n)$	$O(\max(m, t^2))$	$O(t)$			$(m+n)\lVert d\rVert$	$n\lVert k\rVert$	$4\lVert k\rVert$			$2n\lVert s\rVert$
		[60]	$O(mnt)$	$O(mt^2)$	$O(t^2)$			$m(n-1)(t^2-t)\lVert d\rVert/(2t)$	$n\lVert k\rVert$				$n\lVert s\rVert$
	Group 9	[82]	$O(mt)$	$O(mt^2)$	$O(t)$	$O(mt)$		$4m\lVert d\rVert$	$n\lVert k\rVert$	$(t+3)\lVert k\rVert$	$2\lVert k\rVert$		$n\lVert s\rVert$
		[19]	$O(mt)$	$O(mt^2)$	$O(t)$	$O(mt)$		$3m\lVert d\rVert$	$n\lVert k\rVert$	$(t+3)\lVert k\rVert$	$2\lVert k\rVert$		$n\lVert s\rVert$
		[78]	$O(nt)$ if $m \le t$ $O((n+m-t)t)$ otherwise	$O(\max(m^3, mt^2))$	$O(\max(m^2, mt))$			$n\lVert d\rVert$ if $m \le t$ $(n+m-t)\lVert d\rVert$ otherwise	$n\lVert k\rVert$				$n\lVert s\rVert$ if $m \le t$ $(n+m-t)\lVert s\rVert$ otherwise
		[21]	$O(mnt)$	$O(mt^2)$	$O(mt)$			$m(n+1)\lVert d\rVert$	$n\lVert k\rVert$	$2\lVert k\rVert$			$(n+m)\lVert s\rVert$
		[25]	$O(mt^2)$	$O(mt^2)$	$O(mnt)$	$O(m)$		$mt\lVert d\rVert$	$n\lVert k\rVert$	$mt\lVert k\rVert$			$(m+mnt)\lVert s\rVert$
		[16]	$O(\max(nt, m))$	$O(mt^2)$	$O(t^2)$	$O(mt)$		$3m\lVert D\rVert$	$n\lVert k\rVert$	$3\lVert k\rVert$	$\lVert k\rVert$		$t\lVert s\rVert$
	Group 10	[6]	$O(mn)$	$O(mt)$		$O(m)$	$mn\lVert d\rVert/(t-1)$				$nt\lVert k\rVert$	$mn\lVert s\rVert/(t-1)$	
		[5]	$O(mnt)$	$O(mt^2)$		$O(mt)$	$m(n-t+2)\lVert d\rVert$				$(2n+2)\lVert k\rVert$	$\log m(n-t+2)\lVert s\rVert$	
	Group 11	[80]	$O(mt)$	$O(mt^2)$	$O(mt)$		$\frac{mn\lVert d\rVert}{t}$ or $bn\lVert d\rVert$		$nt\lVert k\rVert$			$m\lVert s\rVert$	

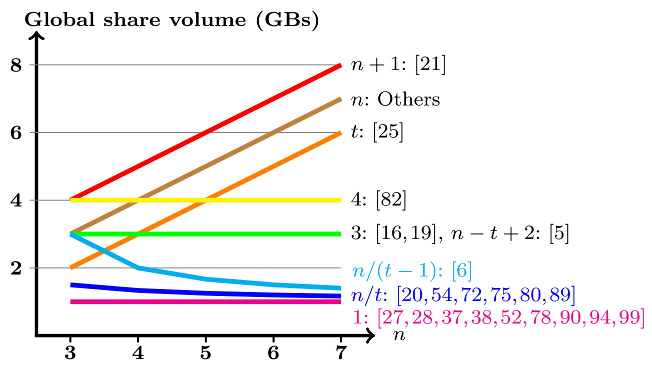


Fig. 11: Global share volume comparison

into three classes. First, [54,72,98] split data before sharing. Hence, share volume is only n/t times that of secrets. However, since the size of shares decreases when t increases, the value of t cannot be bigger than the size of a secret.

Second, global share volumes in [90] and [75,89,20, 80,6] are only 1 and n/t times that of secret data, respectively, because they construct t and n shares, respectively, per data block sizing t secrets.

Third, [37,82,19,16,5] share secrets independently, but they construct fewer than n shares per secret (1, 4, 3, 3 and $n-t+2$ shares, respectively). Hence, share volumes are only 1, 4, 3, 3 and $n-t+2$ times that of secret data, respectively.

Overall, [90,37] require the lowest storage volume (the same as secret data volume) to store shares. However, [90] does not support data availability and [37] supports data availability only when the NB is accessible. Share volumes of [54,72,80,6,5] are a little higher than that of the lowest-share-volume approaches [90,37] if n is close to t, but they do support data availability.

Some SSSs require extra storage to store keys. Most of them use only n or nt keys to share all secrets. Thus, they only consume a small storage volume. However, key volumes of [12,49,25] are greater than the secret data volume (about t^2 [12], $2n$ [49] and t [25] times data volume) because they use different key sets to share a secret. Hence, their overall storage volume (shares, keys and signatures) are greater than that of other SSSs, and thus incurs a higher storage cost.

Finally, all VSSSs and VMSSSs require extra storage to store signatures. The number of signatures is about the number of keys or shares, depending on the verified data type. Thus, overall signature volume is lower than share volume in all VSSSs and VMSSSs. However, if signatures are too small, verification accuracy becomes weak.

Overall, [16] requires the lowest storage volume to store signatures. Hence, its overall storage volume is lower than n times that of secret data. In contrast, [73, 77] require the greatest storage volume to store signatures. Hence, their overall storage volume turn to be greater than other SSSs, i.e., the same as [12,49,25], which construct a huge volume of keys.

4 Frameworks and Architectures for Sharing Secrets in the Cloud

Secret sharing-based cloud frameworks, such as the ones proposed by [84,70], are similar to classical data distribution frameworks [97,69,104] in the cloud and distribute secrets over nodes at a single CSP's (Figure 12). They mostly differ in the SSSs they use. Unlike a classical data distribution framework, such frameworks guarantee data availability by default. Both secret sharing and data reconstruction processes run at a master server's (Figure 13). Although the master server may be a node in the cloud, to reduce privacy breaches in case of hacking, the master server usually stands at the user's side to hide all private parameters and keys from intruders collecting shares.

Two optional verification processes may be enforced by VSSSs. The first process helps verify the correctness of query results at PTs' so that no erroneous query results are transferred back to the master server. The second process runs at the master server's and verifies the correctness of reconstructed query results in case some PTs are not honest.

However, this framework bears a critical security weakness. Since all shares are stored at the same CSPs, if the CSP is hacked, all data can be easily collected and reconstructed by the intruder.

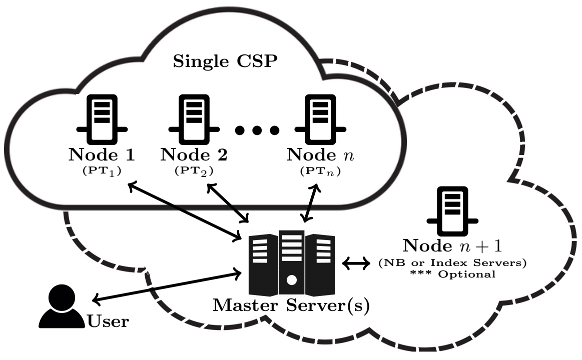


Fig. 12: Architecture from [84,70]

In contrast, the frameworks such as the ones proposed by [6,5,32,63] distribute secrets over multiple CSPs (Figure 14), thus providing better availability (it is unlikely that two or more CSPs all fail at the same

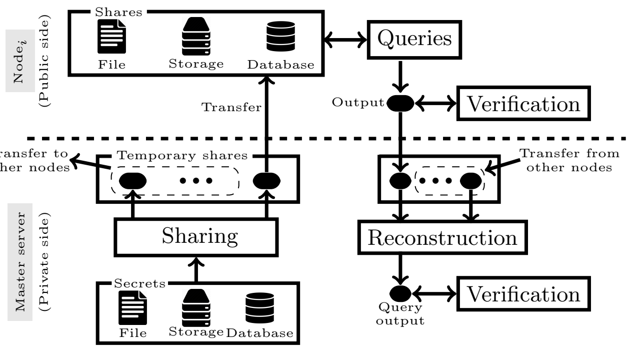


Fig. 13: Cloud SSS framework

time) and privacy (collecting all shares is more difficult than in the one-CSP case).

As in the previous framework, storage and computation costs are still high. However, unlike global data volume, global storage monertary cost might not be n times that of original data because storage cost differs from CSP to CSP. In contrast, data access time is bounded to the slowest CSP. Yet, this problem may be alleviated by both balancing data access time and providing the lowest possible costs [5,4].

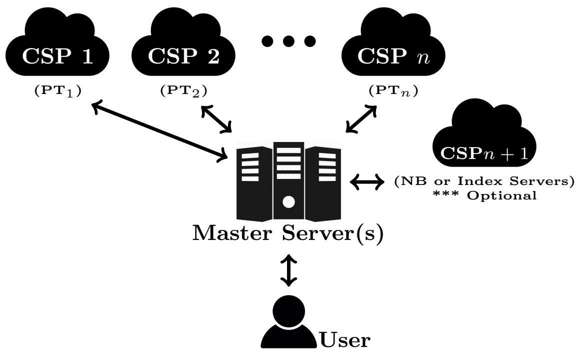


Fig. 14: Architecture from [6,5,32,63]

Finally, an SSSS-based framework [67,66,100,65] generalizes the first two frameworks by distributing secrets over multiple nodes at multiple CSPs' (Figure 15). CSPs play the role of PTs and a number of nodes at CSPs' are the weight of PTs (w_i). Thus, security is not limited by the number of CSPs (n), but by the total number ($w = \sum_i w_i$) of nodes at all CSPs, which can be large. Moreover, shares stored in nodes at any CSPs are not enough to reconstruct any secret since $w_i < t$.

There are some applications, e.g., secure data storage, secure databases and data warehouses, private information retrieval, and data management in the cloud, use the above frameworks.

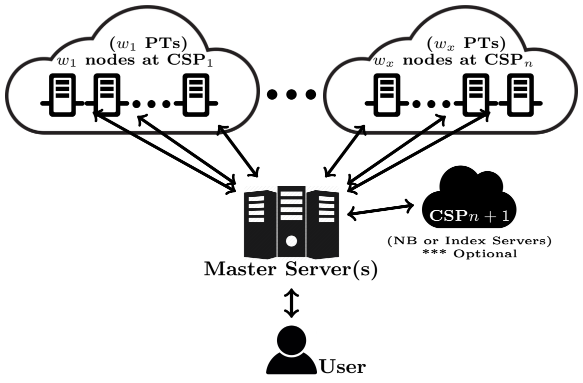


Fig. 15: SSSS-based framework [67,66,100,65]

Table 6: Query types allowed by secret sharing-based cloud applications

Queries	[1]	[36]	[35,44,43,42]	[86]	[88]	[6,5,50]
Update	N	N	Y	Y	Y	Y
Exact match	N	Y	Y	N	Y	Y
Range	N	Y	Y	N	Y	Y
Aggregate	Y	Y	Y	Y	N	Y
Grouping	N	N	N	N	N	Y

Eventually, let us briefly present query functionality in secure data storage solutions for public clouds that use or extend Shamir's SSS [76]. Low-level data storage [79,32] handle pattern search, equijoins and range queries on shares. Table 6 summarizes the querying features of secure cloud databases and data warehouses [35,36,86,1,43,88,44,42,6,5,50].

5 Conclusion

In this final section, we first draw a critical overview of all SSSs surveyed in this paper, including current challenges when using SSSs and in a cloud computing context. Finally, we present some sample applications that can benefit from SSSs.

5.1 Secret Sharing Schemes

Classic SSSs handle data security and availability with high sharing/reconstruction time and storage costs. MSSSs share data at once and reduce both costs. In addition, MSSSs type I support data availability by using a NB, but are vulnerable if the NB is attacked. Hence, to share data with MSSSs type I in the cloud, the NB should be located at a PT's that guarantees high security and availability. In contrast, PSSSs and [5] enhance data privacy by periodically refreshing shares and protecting data from CSP group cheating, respectively.

In addition, VSSSs and VMSSSs can verify the correctness of either or both of data and keys, but these operations induce additional time overhead and require to store signatures in addition to shares. Outer code verification still necessitates to trust PTs, because it is done at PTs'. Moreover, since almost all VMSSSs are also MSSSs type I, their total storage volume (keys, shares and signatures) is still lower than n times that of secret data.

Only [80] verifies the correctness of both data and keys. Although it is an MSSS type II, its total storage volume is only about twice that of secret data. Moreover, its data sharing complexity is also reasonable, i.e., $O(mt)$, while most SSSs have a cubic sharing complexity.

Eventually, only [6,5] verify the correctness of both data and shares. They also minimize global share volume to lower than n times that of secret data. Moreover, [5] can insert new data even though some PTs disappear, i.e., even though some CSPs fail due to technical or economic reasons.

Moreover, PSSSs refresh shares and verify their correctness to improve data privacy. However, computation (to renew shares) and storage (to store signatures) costs induce extra overhead in the refreshment process. Communications to synchronous shares from PTs to PTs are also numerous, thus provoking network bottlenecks.

Some SSSs support features such as updates, search operations, aggregation operations, etc. These features help minimize computation cost at the user's side and reduce communication overhead. Only three SSSs [12, 6,5] support all three operation types: update, exact match and aggregation. However, none can handle composite operations on shares, e.g., simultaneous exact match and aggregation. Performing composite operations on shares remains a challenge in SSSs as of today. Among SSSs that support search and aggregation operations, again only [80,6,5] minimize storage cost. [80] also optimizes data sharing time.

Finally, [53,20] allow the user to add and remove PTs to/from the PT pool. In the cloud, users can thus add and remove CSPs on demand. However, estimating monetary storage cost and detecting attacks or CSP failures is difficult. Thus, taking (or worse, automating) a decision regarding the CSP pool under CSP pricing or privacy constraints is still an open issue.

5.2 Secure Applications in the Cloud

SSSs addressed various issues over time (Section 2). Let us describe below some applications that can benefit from secret sharing for data security.

Textual documents such as emails could be shared with [12,25,6], since these SSSs optimize cost and update and search performance by allowing updates and exact matches directly over shares. Moreover, [25,6] also guarantee data integrity with inner and both inner and outer code verification, respectively. Finally, only [6] optimizes both storage volume and data sharing and reconstruction time.

In databases and data warehouses, update, exact match and aggregation operators are casually used. To optimize query response time, such SSSs as [25,6,5] can be used to leverage cloud databases and warehouses. All these SSSs indeed guarantee data integrity. Moreover, [6,5] also optimize storage cost and [5] allows inserting new data although some CSPs fail. Several secret sharing-based database or warehousing approaches [35, 36,86,1,43,88,44,42,6,5,50] exploit the above-mentioned SSSs.

To handle data streams, SSSs such as [54,82,19,16] can be used, because they optimize sharing time and share secrets independently. Moreover, they require an overall storage volume that is lower than n times that of secret data. [54]'s storage volume is even close to the secret's volume if n and t are big and n is close to t. However, only [82,19,16] guarantee data integrity.

Since memory is still limited in practice, SSSs that share data at once [94,18,37,99,27,28,52,38,78] cannot handle *big data* volumes. However, SSSs that share individual secrets [76,12,3,54,71,72,49,46,61,83,73,23,77, 53,98,82,19,21,25,16,5] or data blocks [90,75,20,89,60, 80,6] allow the execution of the sharing process in main memory or even its parallelization, and thus can share huge data volumes efficiently.

Finally, potential users of database outsourcing should be aware that even frameworks based on normally highly secure SSSs might still be insecure because of inadequate architectural choices or a strong tradeoff in favor of query power [26]. To circumvent this problem, (V)MSSSs that primarily protect multiple secrets are a better choice than (V)SSSs for cloud applications. In any case, users should carefully evaluate the limitation of target SSSs before using them in any applicative context. We hope this survey will help them for this sake.

References

1. Agrawal, D., Abbadi, A.E., Emekci, F., Metwally, A.: Database management as a service: challenges and opportunities. In: 25th IEEE International Conference on Data Engineering (ICDE 2009), Shanghai, China, pp. 1709–1716 (2009)
2. Ali, M., Khan, S.U., Vasilakos, A.V.: Security in cloud computing: Opportunities and challenges. Information sciences **305**, 357–383 (2015)

3. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory **29**(2), 208–210 (1983)
4. Attasena, V.: Secret Sharing Approaches for Secure Data Warehousing and On-Line Analysis Processing in the Cloud. Ph.D. thesis, Université Lumière Lyon 2, France (2015)
5. Attasena, V., Harbi, N., Darmont, J.: fVSS: A New Secure and Cost-Efficient Scheme for Cloud Data Warehouses. In: ACM 17th International Conference on Data Warehouses and OLAP (DOLAP 2014), Shanghai, China, pp. 81–90. ACM (2014)
6. Attasena, V., Harbi, N., Darmont, J.: A Novel Multi-Secret Sharing Approach for Secure Data Warehousing and On-Line Analysis Processing in the Cloud. International Journal of Data Warehousing and Mining **11**(2), 21–42 (2015)
7. Baron, J., Defrawy, K.E., Lampkins, J., Ostrovsky, R.: Communication-optimal proactive secret sharing for dynamic groups. Computer science on applied cryptography and network security **9092**, 23–41 (2016)
8. Beimel, A.: Secret-Sharing Schemes: A Survey. In: 3rd International Conference on Coding and Cryptology (IWCC 2011), Qingdao, China, pp. 11–46 (2011)
9. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The Case of Hashing and Signing. In: 14th Annual International Cryptology Conference (CRYPTO 1994),Santa Barbara, USA, pp. 216–233 (1994)
10. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: 8th Annual International Cryptology Conference (CRYPTO 1988), Santa Barbara, USA, pp. 27–35 (1990)
11. Bilal, K., Malik, S.U.R., Khan, S.U., Zomaya, A.Y.: Trends and challenges in cloud datacenters. IEEE cloud computing **1**(1), 10–20 (2014)
12. Blakley, G.R.: Safeguarding Cryptographic Keys. In: National Computer Conference (AFIPS 1979), Monval, USA, pp. 313–317 (1979)
13. Blakley, G.R., Meadows, C.A.: Security of Ramp Schemes. In: Advances in Cryptology (CRYPTO 1984), Santa Barbara, California, USA, *Lecture Notes in Computer Science*, vol. 196, pp. 242–268 (1985)
14. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. EUROCRYPT 2015 **9057**, 337–367 (2015)
15. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. In: ACM SIGSAC conference on computer and communications security (CCS 2016), Vienna, Austria, pp. 1292–1303 (2016)
16. Bu, S., Yang, R.: Novel and effective multi-secret sharing scheme. In: International Conference on Information Engineering and Applications (IEA 2012), pp. 461–467 (2013)
17. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: the 9th ACM conference on Computer and communications securit (CCS 2002), pp. 88–97 (2002)
18. Chan, C.W., Chang, C.C.: A scheme for threshold multi-secret sharing. Applied Mathematics and Computation **166**(1), 1–14 (2005)
19. Chang, T.Y., Hwang, M.S., Yang, W.P.: An improvement on the lin-wu (t,n) threshold verifiable multi-secret sharing scheme. Applied Mathematics and Computation **163**(1), 169–178 (2005)
20. Chen, G., Wang, H., Wang, L., Jin, Y.: A distributed multi-secret sharing scheme on the (t,n) threshold. In: 2nd International Conference on Network Computing and Information Security (NCIS 2012), Shanghai, China, pp. 358–364 (2012)
21. Chen, W., Long, X., Bai, Y., Gao, X.: A new dynamic threshold secret sharing scheme from bilinear maps. In: International Conference on Parallel Processing Workshops (ICPPW 2007), Xi-An, China, p. 19 (2007)
22. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control. In: 1st ACM Cloud Computing Security Workshop (CCSW 2009), Chicago, USA, pp. 85–90 (2009)
23. Chunming, T., an Yao, Z.: A new (t,n) threshold secret sharing scheme. In: International Conference on Advanced Computer Theory and Engineering (ICACTE 2008), Phuket, Thailand, pp. 920–924 (2008)
24. Cormode, G., Srivastava, D.: Anonymized Data: Generation, Models, Usage. In: 26th IEEE International Conference on Data Engineering (ICDE 2010), Long Beach, USA, pp. 1015–1018 (2010)
25. Das, A., Adhikari, A.: An efficient multi-use multi-secret sharing scheme based on hash function. Applied Mathematics Letters **23**(9), 993–996 (2010)
26. Dautrich, J.L., Ravishankar, C.V.: Security limitations of using secret sharing for data outsourcing. In: IFIP annual conference on data and applications security and privacy (DBSec 2016), Trento, Italy, pp. 151–160 (2016)
27. Dehkordi, M.H., Mashhadi, S.: An efficient threshold verifiable multi-secret sharing. Computer Standards and Interfaces **30**(3), 187–190 (2008)
28. Dehkordi, M.H., Mashhadi, S.: New efficient and practical verifiable multi-secret sharing schemes. Information Sciences **178**(9), 2262–2274 (2008)
29. Derbeko, P., Dolev, S., Gudes, E., Sharma, S.: Security and privacy aspects in mapreduce on clouds: A survey. Computer science review **20**, 1–28 (2016)
30. Ding, C., Pei, D., Salomaa, A.: Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography. World Scientific Publishing, Singapore (1996)
31. Dolev, S., ElDefrawy, K., Lampkins, J., Ostrovsky, R., Yung, M.: Proactive secret sharing with a dishonest majority. Computer science on security and cryptography for networks **9841**, 529–548 (2016)
32. Dolev, S., Li, Y., Sharma, S.: Private and secure secret shared mapreduce (extended abstract). In: IFIP annual conference on data and applications security and privacy XXVI (DBSec 2012), Paris, France, pp. 145–160 (2016)
33. Drgan, C.C., iplea, F.L.: Distributive weighted threshold secret sharing schemes. Information sciences **339**, 85–97 (2016)
34. Dwork, C.: Differential privacy. In: International Colloquium of automata, languages and programming (ICALP 2006), Venice, Italy, pp. 1–12 (2006)
35. Emekci, F., Agrawal, D., Abbadi, A.E.: Abacus: A distributed middleware for privacy preserving data sharing across private data warehouses. In: 6th International Conference on Middleware (USENIX 2005), Grenoble, France, pp. 21–41 (2005)
36. Emekci, F., Agrawal, D., Abbadi, A.E., Gulbeden, A.: Privacy preserving query processing using third parties. In: 22nd IEEE International Conference on Data Engineering (ICDE 2006), Atlanta, USA, pp. 27–37 (2006)
37. Eslami, Z., Ahmadabadi, J.Z.: A verifiable multi-secret sharing scheme based on cellular automata. Information Sciences **180**(15), 2889–2894 (2010)
38. Eslami, Z., Rad, S.K.: A new verifiable multi-secret sharing scheme based on bilinear maps. Wireless Personal Comunications **63**(2), 459–467 (2012)

39. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: 41st annual ACM symposium on Theory of computing (STOC 2009), Bethesda, USA, pp. 169–178 (2009)
40. Gong, G., Harn, L.: Public-key cryptosystems based on cubic finite field extensions. IEEE Transactions on Information Theory **45**(7), 2601–2605 (1999)
41. Gong, G., Harn, L., Wu, H.: The gh public-key cryptosystem. Selected Areas in Cryptography, Spinger **2259**, 284–300 (2001)
42. Hadavi, M.A., Damiani, E., Jalili, R., Cimato, S., Ganjei, Z.: As5: A secure searchable secret sharing scheme for privacy preserving database outsourcing. In: ESORICS DPM/SETOP 2012 International Workshops, Pisa, Italy, pp. 201–216 (2012)
43. Hadavi, M.A., Jalili, R.: Secure data outsourcing based on threshold secret sharing: towards a more practical solution. In: VLDB 2010 PhD Workshop, Singapore, pp. 54–59 (2010)
44. Hadavi, M.A., Noferesti, M., Jalili, R., Damiani, E.: Database as a service: towards a unified solution for security requirements. In: 36th IEEE Annual Conference on Computer Software and Applications Conference Workshops (COMPSACW 2012), Izmir, Turkey, pp. 415–420 (2012)
45. Harn, L., Fuyou, M.: Weighted secret sharing based on the chinese remainder theorem. Network security **16**(6), 420–425 (2014)
46. Harn, L., Lin, C.: Strong (n, t, n) verifiable secret sharing scheme. Information Sciences **180**(16), 3059–3064 (2010)
47. Hashizume, K., Rosado, D.G., Fernndez-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. Internet Services and Applications **4**(1), 1–13 (2013)
48. Hazewinkel, M. (ed.): Totient function. Encyclopedia of Mathematics. Springer, Heidelberg (2001)
49. He, J., Dawson, E.: Multistage secret sharing based on one-way function. Electronics Letters **30**(19), 1591–1592 (1994)
50. He, Z., Wong, W.K., Kao, B., Cheung, D.W.L.: Sdb: A secure query processing system with data interoperability. In: the 41st International Conference on Very Large Data Bases, Hawaii, USA, pp. 1876–1879 (2015)
51. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: 15th Annual international cryptology conference (CRYPTO 1995), Santa Barbara, USA, pp. 339–352 (1995)
52. Hu, C., Liao, X., Cheng, X.: Verifiable multi-secret sharing based on lfsr sequences. Theoretical Computer Science **445**, 52–62 (2012)
53. Hwang, R., Chang, C.: An on-line secret sharing scheme for multi-secrets. Computer Communications **21**(13), 1170–1176 (1998)
54. Iftene, S.: General secret sharing based on the chinese remainder theorem with applications in e-voting. Electronic Notes in Theoretical Computer Science **186**, 67–84 (2007)
55. Iwamoto, M., Yamamoto, H.: Strongly secure ramp secret sharing schemes for general access structures. Information Processing Letters **97**(2), 52–57 (2006)
56. Joshi, J.B., Takabi, H., Ahn, G.J.: Security and privacy challenges in cloud computing environments. IEEE Security and Privacy **8**, 24–31 (2010)
57. Khan, M.A.: A survey of security issues for cloud computing. Journal of network and computer applications **71**, 11–29 (2016)
58. Kher, V., Kim, Y.: Securing distributed storage: challenges, techniques, and systems. In: ACM workshop on Storage security and survivability (StorageSS 2005), Fairfax, USA, pp. 9–25 (2005)
59. Komargodski, I., Zhandry, M.: Cutting-edge cryptography through the lens of secret sharing. EUROCRYPT 2015 **9563**, 449–479 (2015)
60. Li, S., Lai, H., Wu, W., Jiang, S.: Novel space efficient secret sharing for implicit data security. In: 8th International Conference on Information Science and Digital Content Technology (ICIDT 2012), Jeju, Japan, pp. 283–286 (2009)
61. Liu, Y.X., Harn, L., Yang, C.N., Zhang, Y.Q.: Efficient (n, t, n) secret sharing schemes. Journal of Systems and Software **85**(6), 1325–1332 (2012)
62. Mashhadi, S.: Secure publicly verifiable and proactive secret sharing schemes with general access structure. Information sciences **378**, 99–108 (2017)
63. M.Muhil, Krishna, U., Kumar, R., Anita, E.A.M.: Securing multi-cloud using secret sharing algorithm. In: IFIP annual conference on application of fuzzy systems and soft computing (ICAFS 2016), Vienna, Austria, pp. 145–160 (2016)
64. Morillo, P., Padr, C., Sez, G., Villar, J.: Weighted threshold secret sharing schemes. Information processing letters **70**(5), 211–216 (1999)
65. Nojoumian, M., Stinson, D.R.: Social secret sharing in cloud computing using a new trust function. In: 10th Annual International Conference on Privacy, Security and Trust (PST 2012), Paris, france, pp. 161–167 (2012)
66. Nojoumian, M., Stinson, D.R.: Socio-rational secret sharing as a new direction in rational cryptography. In: 3rd International Conference Decision and Game Theory for Security (GameSec 2012), Budapest, Hungary, pp. 18–37 (2012)
67. Nojoumian, M., Stinson, D.R., Grainger, M.: Unconditionally secure social secret sharing scheme. Information security, IET **4**(4), 202–211 (2010)
68. Oktay, K.Y., Mehrotra, S., Khadilkar, V., Kantarcioglu, M.: Semrod: Secure and efficient mapreduce over hybrid clouds. In: International Conference on Management of Data (SIGMOD 2015), Melbourne, Australia, pp. 153–166 (2015)
69. Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A survey of data replication techniques for mobile ad hoc network databases. The VLDB Journal **17**(5), 1143–1164 (2008)
70. Pal, D., Khethavath, P., Thomas, J.P.: Multilevel threshold secret sharing in distributed cloud. Security in Computing and Communications **536**, 13–23 (2015)
71. Parakh, A., Kak, S.: Online data storage using implicit security. Information Sciences **179**(19), 3323–3331 (2009)
72. Parakh, A., Kak, S.: Space efficient secret sharing for implicit data security. Information Sciences **181**(2), 335–341 (2011)
73. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT 1991), Brighton, UK, pp. 522–526 (1991)
74. Perlner, R.A., Cooper, D.A.: Quantum resistant public key cryptography: a survey. In: 8th Symposium on Identity and Trust on the Internet (IDtrust 2009), Gaithersburg, Maryland, USA, ACM ICPS, pp. 85–93 (2009)
75. Runhual, S., Liusheng, H., yonglong, L., Hong2, Z.: A threshold multi-secret sharing scheme. In: IEEE International Conference on Networking, Sensing and Control (ICNSC 2008), Sanya, China, pp. 1705–1707 (2008)

76. Shamir, A.: How to Share a Secret. Communications of the ACM **22**(11), 612–613 (1979)
77. ShanYue, B., Hong, Z.: A secret sharing scheme based on ntru algorithm. In: Wireless Communications, Networking and Mobile Computing (WiCom 2009), Beijing, china, pp. 1–4 (2009)
78. Shao, J., Cao, Z.: A new efficient (t,n) verifiable multi-secret sharing (vmss) based on ych scheme. Applied Mathematics and Computation **168**(1), 135–140 (2005)
79. Shen, F., andHai Jiang, C.M., Xu, Z.: Towards secure and reliable data storage with multi-coefficient secret sharing. In: 2010 IEEE 10th International Conference on Computer and Information Technology (CIT), Bradford, UK, pp. 797–802 (2010)
80. Shi, R., Zhong, H., Huang, L.: A (t, n)-threshold verified multi-secret sharing scheme based on ECDLP. In: International Conference on InterSoftware Engineering, Artificial Intelligence, Networking, and Parallel/Distributed (ACIS 2007), pp. 9–13 (2007)
81. Sion, R.: Secure data outsourcing. In: 33rd international conference on Very large data bases (VLDB 2007), Vienna, Austria, pp. 1431–1432 (2007)
82. T.-Y.Lin, T.-C.Wu: (t,n) threshold verifiable multi secret sharing scheme based on factorisation intractability and discrete logarithm modulo a composite problems. IEEE Computer and Digital Techniques **146**(5), 264–263 (1999)
83. Takahashi, S., Iwamura, K.: Secret sharing scheme suitable for cloud computing. In: 27th international conference on advanced information networking and applications (AINA 2013), Barcelona, Spain, pp. 530–536 (2013)
84. Takahashi, S., Iwamura, K.: Secret sharing scheme suitable for cloud computing. In: International conference on advanced information networking andapplications (AINA 2013), Barcelona, Spain, pp. 530–537 (2013)
85. Tang, C., an Yao, Z.: Definition and construction of multi-prover zero-knowledge arguments. In: International conference on communications and mobile computing (CMC 2009), Yunnan, China, pp. 375–379 (2009)
86. Thompson, B., Haber, S., Horne, W.G., Sander, T., Yao, D.: Privacy-preserving computation and verification of aggregate queries on outsourced databases. In: 9th International Symposium on Privacy Enhancing Technologies (PETS 2009), Seattle, USA, pp. 185–201 (2009)
87. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: 14th European Conference on Research in Computer Security (ESORICS 2009), Saint-Malo, France, pp. 355–370 (2009)
88. Wang, S., Agrawal, D., Abbadi, A.E.: A comprehensive framework for secure query processing on relational data in the cloud. In: 8th VLDB International Conference on Secure Data Management (SDM 2011), Berlin, Germany, pp. 52–69 (2011)
89. Wang, S.J., Tsai, Y.R., Shen, C.C.: Verifiable threshold scheme in multi-secret sharing distributions upon extensions of ecc. Wireless Personal Comunications **56**(1), 173–182 (2011)
90. Waseda, A., Soshi, M.: Consideration for multi-threshold multi-secret sharing schemes. In: 2012 International Symposium on Information Theory and its Applications (ISITA 2012), Honolulu, USA, pp. 265–269 (2012)
91. Wei, D.S., Murugesan, S., Kuo, S.Y., Naik, K., Krizanc, D.: Enhancing Data Integrity and Privacy in the Cloud: An Agenda. IEEE Computer **46**(11), 87–90 (2013)
92. Wong, T.M., Wang, C., Wing, J.M.: Verifiable secret redistribution for archive systems. In: International IEEE Security in Storage Workshop, pp. 94–106 (2002)
93. Xu, Z., Martin, K., Kotnik, C.: A Survey of Security Services and Techniques in Distributed Storage Systems. In: 2011 International Conference on Security and Management (SAM 2011), Las Vegas, USA, pp. 3–9 (2011)
94. Yang, C.C., Chang, T.Y., Hwang, M.S.: A (t,n) multi-secret sharing scheme. Applied Mathematics and Computation **151**(2), 483–490 (2004)
95. Yao, A.C.C.: How to generate and exchange secrets. In: 27th annual symposium on foundations of computer science, Toronto, Canada, pp. 162–167 (1986)
96. Zhang, C., Chang, E.C., Yap, R.H.: Tagged-mapreduce: A general framework for secure computing with mixed-sensitivity data on hybrid clouds. In: International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014), Chicago, IL, USA, pp. 31–40 (2014)
97. Zhang, F., Chen, H.: Security-preserving live migration of virtual machines in the cloud. Network and systems management **21**(4), 562–587 (2013)
98. Zhao, D., Peng, H., Wang, C., Yang, Y.: A secret sharing scheme with a short share realizing the (t,n) threshold and the adversary structure. Computers and Mathematics with Applications **64**(4), 611615 (2012)
99. Zhao, J., Zhang, J., Zhao, R.: A practical verifiable multi-secret sharing scheme. Computer Standards and Interfaces **29**(1), 138–141 (2007)
100. Zheng, T., Wu, H., Lin, H.W., Pan, J.: Application of belief learning model based socio-rational secret sharing scheme on cloud storage. In: 6th International Conference on Genetic and Evolutionary Computing (ICGEC 2012), Kitakyushu, Japan, pp. 15–18 (2012)
101. Zhou, L., Schneider, F.B., Renesse, R.V.: Apss: proactive secret sharing in asynchronous systems. ACM transactions on information and system security (TISSEC) **8**(3), 259–286 (2005)
102. Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A.: Security and privacy in cloud computing: A survey. In: International conference on semantics, knowledge and grids (SKG 2010), Beijing, China, pp. 105–112 (2010)
103. Zhou, Z., Zhang, H., Du, X., Li, P., Yu, X.: Prometheus: Privacy-aware data retrieval on hybrid cloud. In: Proceedings of the IEEE INFOCOM 2013, Turin, Italy, pp. 2643–2651 (2013)
104. Zissis, D., Lekkas, D.: Addressing cloud computing security issues. Future generation computer systems **28**(3), 583–592 (2012)