Arab J Sci Eng (2017) 42:873–883
DOI 10.1007/s13369-016-2357-2

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Secure Data Storage in Cloud: An e-Stream Cipher-Based Secure and Dynamic Updation Policy

Dharavath Ramesh[1] · Rahul Mishra[1] · Damodar Reddy Edla[2]

Received: 16 June 2016 / Accepted: 17 November 2016 / Published online: 2 December 2016

Abstract With the growth of www, cloud computing paradigm has become a massive computing environment as a pay-per-use model to the user with shared pool of resources. It also provides on time demand services from anywhere anytime as instant updating and elasticity with measured services. Along with these services, cloud model has major security concerns as securing crucial data of user and maintaining integrity of VM's disk for consistent updation and retrieval from different sources. To achieve instant and secure migration of VMs, cloud service provider has introduced a concept of virtualization to maintain virtualized servers, OS, storage, etc. To achieve secure virtualization and dynamic updation of data in VM's disks, we introduce a model named secure e-stream cipher-based encryption/decryption as ChaCha20 method for maintaining proper security to the user's sensitive data at cloud data center. To maintain proper integrity and authenticity between VM's disks, a new methodology named dynamic version of dynamic Merkle hash B+ tree (DMBHT) with q-SDH secure short signature without random oracle signature scheme has been introduced. This scheme has efficient rate as ϕ erasure code (Tornado-z code) for forming block tag at leaf level of DMBHT. Our proposed methodology along with DMBHT has worst case complexity as $O(\log n)$ instead of $O(n)$ and has better public auditability to attain expeditious and secure modifications with proper updations.

Keywords Secure virtualization · Cloud computing · DMBHT · q-SDH short signature scheme without random oracle

✉ Dharavath Ramesh
ramesh.d.in@ieee.org

Rahul Mishra
mishrar93@yahoo.com

Damodar Reddy Edla
dr.reddy@nitgoa.ac.in

1 Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad, Jharkhand 826004, India

2 Department of Computer Science and Engineering, National Institute of Technology, Farmagudi, Goa 403401, India

1 Introduction

Cloud paradigm is defined as a computing environment for facilitating pervasive on demand, anytime, and anywhere network access from a shared pool of resources (storage, servers, networks, etc.) [1]. It also conceptualizes a new posterity information technology [2] construction for the business world which offers instant modification with updation, better resource utilization, low-cost computation, and better storage platform to use IT. Apart from the benefits, cloud model has lot of major security concerns with respect to user's sensitive outsourced data such as secure data storage at data center and proper integrity. Cloud model's service is further divided into software-as-a-service (SaaS; a third party-based service like e-mail, Google apps, health-related information apps, etc.), platform-as-a-service (PaaS; provides a platform to the developer for development of customized applications like Apprenda, etc.), Iaas (infrastructure-as-a-service; provides an environment for auditing and integrates cloud model services). Nowadays, cloud service provider offers service in the form of virtualization [3,4]. Virtualization environment creates a model with virtualized server, storage, network, OS, etc. Virtualization model also improves flexibility and agility. With virtualization, cloud service provider (CSP) deploys workloads to other system instead of one to achieve performance and availability. In spite of these merits, virtualization environment has major security concerns related to VM's disk and secure VM migration.

In the cloud environment, CSP offers new methodologies as dynamic and secure updation to attain proper authenticity, availability, retrievability, and rapid updation with modification [5,6]. Some existing methodologies perform updation and modification operations based on a static mechanism for updation. In the current scenario, the growth of the data is increasing time by time in a large quantity and users daily upgrade their data in frequent time frames, CSP need to maintain rapid and instant modification of the data. In order to accommodate this instance, in this manuscript, we discuss dynamic PoR public auditable scheme to attain dynamic updation for user's data. Our scheme is based on dynamic version of MBHT (Merkle B+ hash tree) with secure short signature scheme for proper authenticity. It also have super secure e-stream cipher-based ChaCha20 encryption and decryption scheme. The road map of our scheme is as follows:

- First, we define a super secure e-stream cipher-based encryption and decryption scheme for maintaining security.
- Second, dynamic form of MBHT is introduced to maintain large hash (SHA-512) values of VM's disk at leaf level with q-SDH secure short signature without random oracle method to maintain proper authenticity.

2 Related Works

In the literature, many researchers have exemplified much of evolving significance in framework of distantly stored data authenticity with proper validity and verification with integrity. In the recent literature, Pearson [7], Gu and Cheung [8], Siebenlist [9] have discussed some methodologies on virtualization for data cloud center in cloud model. Takabi et al. [10] proposed an architecture of VM's disk security at data center by using block cipher-based encryption and decryption method for attending proper security and MHT for maintaining proper validating scheme along with Merkle's one time signature scheme to attain authenticity. This is secure than previously defined signature schemes. But this method suffers from some security-related issues such as cryptanalysis, IND–CPA, and collision attacks. Further, Wang [11] and Hay et al. [12] explored a brief discussion on VM's disk-related security concerns. To attain proper security and integrity, a method named Cloud Visor [13] based on AES-CBC 128 bit with classical MHT and MD5 hash function is introduced. But, this method has some security concerns such as differential cryptanalysis of AES-CBC 128 bit.

Shacham and Waters [14] characterized a latest version of public auditable scheme which is based on secure (q-SDH) BLS signature scheme for attaining proper authenticity and also supports mutual authentication. But BLS signature scheme has two step time-consuming operations during signature verification and not suitable efficient for dynamic updation or modification of data at data center. Wang et al. [15] proposed a similar scheme of BLS signature scheme along with classical MHT-based PoR to achieve rapid modification or updation and proper integrity by signing at root (R) of MHT by $Sign_{\mathrm{sup}\, r\_key}(R)$. Boneh et al. [16] constructed a scheme called a secure short signature which has better and faster performance over RSA-based scheme.

A new dynamic auditing protocol was introduced by Chris [17] as TPA (third party auditor) that checks integrity and authenticity of sensitive data of user at cloud data center. But due to direct involvement of TPA in security checks, there will be a chance of leaking of crucial data. Further, some other researchers were given description on rate ϕ erasure code for encoding and decoding with proper retrievability of complete data during downloading [18–23]. Most of these methodologies are based on Reed–Solomon code rate ρ erasure code or error correcting code. But due to some major issues related to time computation in encoding and decoding for large data blocks, this scheme was dropped and replaced by some other efficient methodologies. In keeping the above constraints, in this manuscript, we propose a secure e-stream cipher-based encryption/decryption as ChaCha20 method for maintaining proper security to the user's sensitive data at cloud data center. To maintain proper integrity and authenticity between VM's disks, we introduce a dynamic version of DMBHT (dynamic Merkle hash B+ tree) with q-SDH secure short signature without random oracle signature scheme.

3 Methodology

The objective of this methodology is to maintain proper secrecy and translucent integrity for virtual machine's disk under feasible virtual paradigm and dynamic updation of data at cloud data center. In our proposed scheme, we described a member of SALSA e-stream cipher family named ChaCha20 [24,25] stream cipher to achieve appropriate privacy for sensitive data stored at VM disk. And a dynamic version of Merkle B+ hash tree along with unforgeable signature scheme as short signature without random oracle is used to maintain proper integrity for VM's disk.

3.1 Stream Cipher-Based Encryption (ChaCha20) Method

In this section, we brief about our encryption method based on stream cipher ChaCha20 which is a member of SALSA e-stream cipher family. The elementary part of the stream cipher is *keystream setup*, *encryption phase*, and *decryption phase*.

Constant	**Constant**	**Constant**	**Constant**
Key	**key**	**key**	**key**
Key	**key**	**key**	**key**
Counter	**counter**	**nonce**	**nonce**

Fig. 1 Initial state matrix

a+=b	**d^=a**	**d<<<=16**
c+=d	**b^=c**	**b<<<=12**
a+=b	**d^=a**	**d<<<=8**
c+=d	**b^=c**	**b<<<=7**

Fig. 2 Round function operation

3.1.1 Key Setup

In key setup phase, basic inputs are 256-bit initial keys along with 96-bit nonce and 32-bit counter. These values are formed by performing concatenation of 8-bit little endian integers. We arrange these values into initial state matrix and apply quarter-round function operation on the matrix. The corresponding initial state matrix and round function operation are described in Figs. 1 and 2.

Initially, we apply 20 rounds on initial original state matrix by performing transformation of column and diagonal matrix on every 128-bit of new input plaintext. Each word updates twice, and quarter-round function is applied. To update the values of state matrix, this functionality has been applied as column-wise and diagonal-wise operation. The notation of the quarter function is represented as *QUARTER_ROUND(a,b,c,d)*, where *a, b, c, and d* are the indexes of the *ChaCha20* which is viewed as a vector. The textual representation of the quarter-round function and description of round operation on initial state matrix along with test vector is stated as follows;

```
INNER_BLOCK(State_matrix):
QUARTER_ROUND(state_matrix, 0, 4, 8, 12)
QUARTER_ROUND(state_matrix,1,5, 9, 13)
QUARTER_ROUND(state_matrix,2, 6, 10, 14)
QUARTER_ROUND(state_matrix,3, 7, 11, 15)
QUARTER_ROUND(state_matrix,0, 5, 10, 15)
QUARTER_ROUND(state_matrix,1, 6, 11, 12)
QUARTER_ROUND(state_matrix,2, 7, 8, 13)
QUARTER_ROUND(state_matrix,3, 4, 9, 14)
END.
```

Description of round_operation on initial state matrix

***a**	**1**	**2**	**3**
***b**	**5**	**6**	**7**
***c**	**9**	**10**	**11**
***d**	**13**	**14**	**15**

As an example, if QUARTER_ROUND function applies on (0, 4, 8, 12) to initial state matrix, then quarter-round operation updates the values marked with pointer symbol. As a result, key setup phase produces output as 64 random bytes.

3.1.2 Encryption Phase

In this phase, to encrypt the key, a 256-bit key stream is taken and 20 rounds of quarter-round function on initial state matrix *XORed* have been performed with 16 words plaintext_file. But the size of plaintext_file is varied in size and extra bit of key stream is discarded in case of less size of plaintext_file. Working of encryption with ChaCha20 is depicted in Fig. 3.

The pseudocode of the encryption process is shown below.

Algorithm of Encryption Process:

```
1.  Algo_encryption_keygen (Key, Counter ,Nonce, plaintext_file)
2.      for counter = 1 to (cel (len (plaintext_file) / 64))
3.          key_stream = algo_block (Key, Counter, Nonce)
4.          block = plaintext_file [(counter*64) ---------------------(counter*64 - 1)]
5.          ciphertext + = block ⊕ key_stream
6.      end
7.          if ((cel (plaintext_file % 64)! = 0))
8.            key_stream = algo_block (key, counter, nonce)
9.            block = plaintext_file [(counter*64) --------------------- len (plaintext_file) - 1)
10.       ciphertext + = (block ⊕ key_stream)[0 ------------------ len (plaintext_file) %64)]
11.         end
12. return ciphertext
13.   end
```

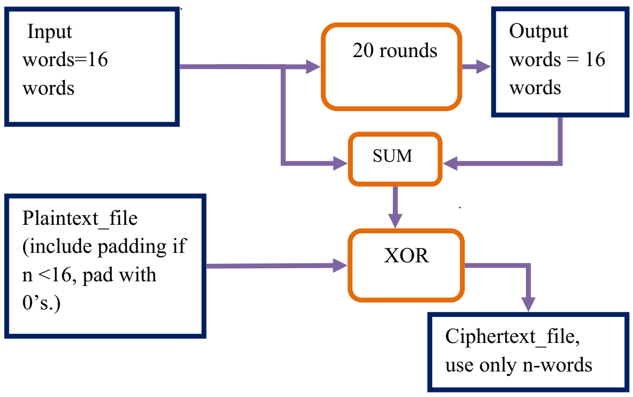
Input words=16 words
20 rounds
Output words = 16 words
SUM
Plaintext_file (include padding if n <16, pad with 0's.)
XOR
Ciphertext_file, use only n-words

Fig. 3 Working of ChaCha20 stream cipher

3.1.3 Decryption Phase

Decryption operation of e-stream cipher is same as encryption operation by performing *XOR* operation on ciphertext_file with a key stream. To maintain proper integrity between different VM's disks, an improved version of Merkle B+ hash tree (DMBHT) is used. The representation of DMBHT is presented in Sect. 3.2.

3.2 Dynamic Merkle B+ Hash Tree (DMBHT)

In the present era of cloud computing environment, clients prefer to store their data to cloud data center. Cloud service provider (CSP) stores this information in the form of metadata in a small fraction of located area. For the usage purpose, any client or user can retrieve the data from the allocated portion. To perform this activity, a scheme called DMBHT is used [26]. In our proposed model, we describe a dynamic variant of Merkle B+ hash tree. To perform this, we arrange number of hash values $H(M_1), H(M_2), \ldots H(M_n)$ (*SHA-512 hash function) [27] at the leaf level of the tree. This construction has worst case complexity as $O(\log n)$. Each leaf node of DMBHT has parts of left(t), right(t), and rank(t), i.e.,{* rank shows total number of dependent descents on one node}.

$$\rho(t) = \left\{ \begin{array}{l} 0,\ \text{if t has at most 2 child nodes} \\ 1,\ \text{if t has more than 2 child nodes} \end{array} \right\}$$

$U(elem) = h(H(M))$, calculate the value of

$$U(t) = h[U(left)||U(right)||U(middle)||\rho(t)||rank(t)].$$

To maintain proper authenticity between client and server, we introduce a scheme called short signature without random oracle by signing at root node (R) of DMBHT using private key generated in key generation method $[Sign_{prvt\_key}(U(W))]$ and finally deploy the data file along with signature (φ), DMBHT and U(W) to cloud data center.

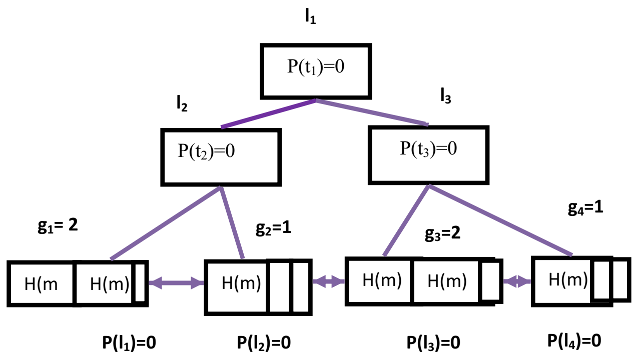

Fig. 4 Description of DBMHT

This instance is depicted in Fig. 4. Our construction includes the following parameters.

Key_generation (λ^k) **→(pub_key,prvt_key)**: This phase of algorithm takes a security token λ as input and produce pair of public key and private keys. These keys are used during updation and signing.

Preparation (prvt_key,FB,FB$_{tag}$)→(φ,$Sign_{prvt\_key}(U(W))$, DMBHT): This phase of algorithm generates block tags (FB) by using rate ϕ erasure code tornado-z code for encoding of data files in blocks and form hash values of these block tags as (FB$_{\text{tags}}$)$FB_{tag} = \{H(M_i)\}$at leaf level of DMBHT (M_i) for $0 \le i \le n$. And also it produces set of signatures φ_i at non-leaf nodes of DMBHT. Finally, it signs at root node (R) as an output of this step.

Challenge_generation (l) → Qr: In this step of algorithm, client generates set of quires and a random value $s_i \in Z_p$ for each index of data file block which are arranged at leaf level of dynamic tree. And also it produces IDs for all data file block tag set defined as ((ID = id$_1$, id2. . . . . . .id$_k$)). Finally, client sends computed value of (Qr_i, s_i) as challenges to server side for verification step of block tag at server side.

Proof Generation (Qr**,DMBHT,FB,FB$_{\text{tag}}$,** φ)**→Prf**: In this phase of algorithm, server generate proofs (Prf) of challenges on behalf of challenges generated by client. Server has values (DMBHT,FB,FB$_{\text{tags}}$, φ). So finally, it produce proofs (Prfs) to authenticate the privacy of queries.

$\omega = \sum_{i=1}^{i_k} s_i m_i$ and $V = \prod_{i=i_1}^{i_k} V_i r_i$ on complete verification at server side and sends block tag sets $FB_{tag} = \{H(M_i)\}$ to client/users.

Verification: In this phase of proposed algorithm, client runs verification function to check validity with authenticity of VM disk's data. During verification phase, client generates $Sign_{prvt\_key}(U(W))$ value at root R. For validation, VM's disk data are stored at leaf level by checking $e(\delta, \delta) = e(kl^m g^s, g)$.

$$\begin{aligned} e(\delta, \delta) &= e\left(g^{\sqrt{(u+mv+s)}}, g^{\sqrt{(u+mv+s)}}\right) \\ &= e(g, g)^{u+mv+s} \end{aligned}$$

$$= e(g^{u+mv+s}, g)$$
$$= e(kl^m g^s, g)$$

3.2.1 Updation Process

Suppose any user among K wants to update their data from M to $M^!$ stored at kth block in VM's disk at leaf level of DMBHT, user requests for update with UPDATE _REQUEST() where,

$update\_request \epsilon \{insertion, modification, deletion\}$

For the updation purpose, user sends update request $\{M, M_k^!, i, \varphi_k^!\}$ to the server. After receiving updation request from client side, service provider updates value of M to $M^!$ and $H(M_k)$ with $H(M_K^!)$ in DMBHT by generating a new root value R' and forms new proofs (Prnew) as Pr$_{new}$ = $\{\varphi^!, H(M_K^!), Sign_{prvt\_key}(H(R^!)), R^!\}$. Now server has two values of proofs {Prold, Prnew} and sends these values to client. On receiving the value of proofs from server side, client verifies these proofs by executing UPDATE_VERIFICATION() function and generates output (True, False). The insertion, updation, and deletion operations of DMBHT are shown in Fig. 5.

3.3 Signature Scheme

To attain proper authentication and integrity, we introduce a q-strong short signature scheme(SSO) [28] at root of tree (R), whereas previous existing methods work based on 2 step time-consuming verification operation named as BLS signature scheme. The methodology of q-SDH is intractable iff there is no *T-time* algorithm to solve *q-SDH* problem in *at least* $\in$*time*. This signature scheme is *q-SDH* and unforgeable intractable in nature. This *q-SDH* problem defined as (G_1, G_2)groups, given (p+2) tuples $(g_1, g_2, g_2^a, g_2^{a^2}, \ldots g_2^{a^p})$ as we have inputs as $g_1 = \psi(g_2)$ and output as$(s, g^{1/a+s})^2$, $s \in Z_p^*$. Now let e: G * G → G$_T$ where G, G$_T$ are two multiplicative cyclic groups having generator g which is defined as bilinear pairing {*bilinear pairing defines as –participating pairs of element of groups* G_1*and* G_2*to elements of group* G_T *but* $|G_1| = |G_2| = |G|$}. The properties of pairing are defined as;

i. Degenerate: $e(P, Q) \neq 1$
ii. Bilinearity: $\forall_{a,b} \in F_p^*, \forall R \in G_1, \forall S \in G_2 \quad e(aR, bS) = e(R, S)^{ab}$

Finally, SSO signature scheme has security parameters (G, G$_T$, e, p, g, Z$_p$[+1]) where Z$_p$[+1] is defined as {a ∈ Z$_p$| a is a quadratic residue mod p} and signed message $m \in Z_p$.

Key_generation: Selects two random values $u, v \in_R Z_p^*$ and calculates $k = g^u, l = g^v$. Finally, computes public key (k, l) and private key(u, v).

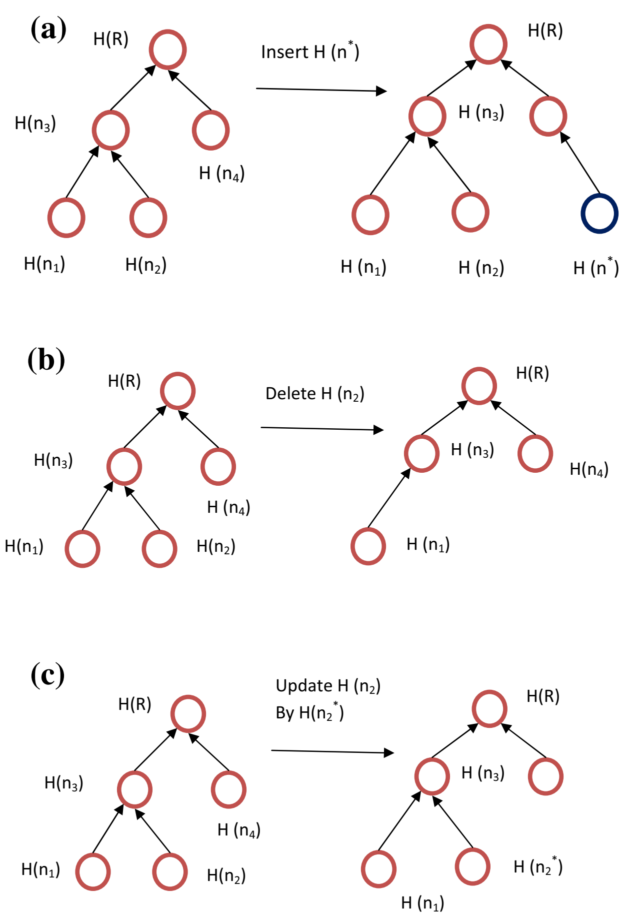

Fig. 5 Insertion, updation, and deletion operation in DMBHT. **a** Insertion operation on DMBHT. **b** Updation operation on DMBHT. **c** Deletion operation on DMBHT

Signing: Given (k, l) and (u, v) along with the message $m \in Z_p$ and also selects random value $s_i \in Z_p^*$ and calculates $\delta = g^{(u+mv+s)^{1/2}} \in G$. The function, $\sqrt{u + mv + s}$ calculates over modulo$p$ but whenever this value is not quadratic residue modulo p then again find next random value of s and finally compute signature (δ, s).

Verification: Given public key (G, G$_T$, p, g, l, k) message $m \in Z_p$ and sign(δ, s) and verifies $e(\delta, \delta) = e(kl^m g^s, g)$ and $e(\delta, \delta) \doteqdot e(g^{\sqrt{(u+mv+s)}}, g^{\sqrt{(u+mv+s)}})$. For maintaining proper integrity, we use SHA-512 hash function to form hash value of VM's disk at leaf level of DMBHT. SHA-512 has many advantages over SHA-256 and MD5–SHA. It has 37.5% less round cycles per byte on 64 bit machine and also more secure as with 80 rounds operation on 128 byte block which helps to achieve better security on 64 bit ALU's operation.

3.4 Working of VM's Disk Data Security and Integrity

As per the proposed methodology, we store encrypted sensitive data in VM's disk and create DMBHT to arrange large

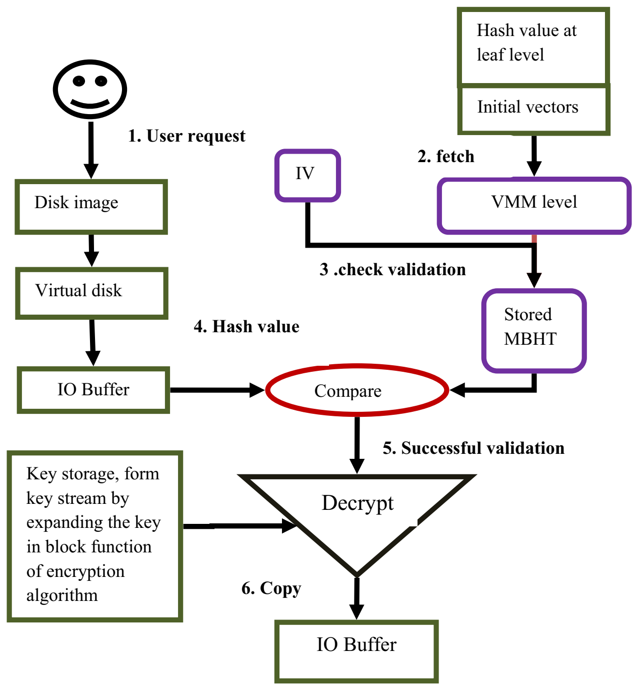

Fig. 6 Working procedure of our model

hash values of VM's disk at its leaf level. An unforgeable signature scheme *SSO* is used to sign at all non-leaf nodes and attain signature (δ) at root R of DMBHT. This procedure is represented as follows.

i. Whenever a user/client wants to upload their data at data center, then the data are encrypted by using e-stream cipher ChaCha20.
ii. After that find hash values of VM's disk and arrange large number of hash values into DMBHT.
iii. Finally, send DMBHT along with signature value to server.
iv. At the time of downloading, first checks the integrity and authenticity of VM's disk, then decrypt the data stored at VM's disk by attending initial vector stored at VMM level in little ending format. The complete procedure of downloading of data from cloud data center is depicted in Fig. 6.

4 Security Analysis

4.1 Security Analysis of Stream Cipher: ChaCha20

In our construction, we discussed about secure and faster e-stream cipher for providing security for the sensitive data of a client at cloud data center. Based on the methodology presented in some other works [29,30], we describe security analysis of ChaCha20. In our construction, we mentioned about a key stream of 256-bits generated by block function $\{0,1\}^{256} \times \{0,1\}^{128} \rightarrow \{0,1\}^{512}$. This function compiles 16 bytes as inputs and 64 bytes as output. The encryption method secures the data from IND-CPA and IND-CTXT attacks (integrity of CTs) [29]. We also discuss about an adversary model A which has two notations as encryption oracle and decryption oracle. In their adversary model, adversary has right to generate at most y queries for both oracle model. This model has series of games G_i over encryption $\mathrm{E}^!$ and decryption $D^!$ to proof attack by adversary. These security concerns are described as follows.

IND-CPA security: If an adversary is capable to distinguish between (C, T) which is generated by initial encryption E^0 from random bit strings of same length, then the adversary wins the game. So probability of adversary's advantages to win the game is;

$$Adv^{cc}_{IND\varsigma-CPA} = \left|\Pr[A^{\mathrm{E}_0} \rightarrow 1]\right| - \Pr[A^{\xi} \rightarrow 1]$$

IND-CTXT Security: If an adversary is capable to forge ciphertext by changing output tuple (N, A, C, T) to D_k $(N, A, C, T) = (N, A, P) \neq \perp$, where (N, A, C, T) may not be the proper ciphertext set by encryption oracle and decryption oracle output $\perp$ which is not valid, then it will show that adversary may change CT values. So the probability of adversary's advantages to win the game is;

$$Adv^{cc}_{IND-CPA} = \left|\Pr[A^{\mathrm{E}_0, D^0} forges]\right|$$

4.2 Security Analysis with Hash Function SHA-512

In this section, we discuss about security analysis with hash function. In previously existing methods, SHA-256 hash function is used to attain proper integrity of VM's but SHA-256 has some security concerns such as multicollision attacks [31], Chabaud's attack, and many more attacks. Here, we discuss about multicollision attack on SHA-512 hash function which is secured from Dobberth's, multicollision, and Chabaud's attacks. As compared to SHA-256, SHA-512 hash function is faster and having strong diffusion of message block with low weight difference.

Multicollision attack: Multicollision attack is also referred as Joux's simple multicollision attack on iterated hash function. Joux's calculates esteemed complexity for collision $\Theta(s.2^{n/2})$ in any iterated hash function, whereas birthday paradox's attack complexity is too high $\Theta(2^s \times 2^{n(2^s-1}/2^s)$. The basic concept behind Joux's method to calculate d-successive collisions in hash function is defined as follows.

Let us assume for $X, X^! \in \{0,1\}^n$ and $b \in \{0,1\}^t$. Now we describe compression function by using these notations $X \xrightarrow{b} X^!$ where compression $(X, b) = X^!(|X| = \left|X^!\right| = n)$ and

$|b| = t$. For multiple message blocks, we further enhance attack instance in the below manner.

$$X \xrightarrow{b_1 b_2 \ldots\ldots b_n} X^!$$

$$X_0 \xrightarrow{b_1^1} X_1 \text{ and } X_0 \xrightarrow{b_1^2} X_1$$

$$X_1 \xrightarrow{b_1^1} X_2 \text{ and } X_1 \xrightarrow{b_1^2} X_2$$

.

.

$X_{s-1} \xrightarrow{b_s^1} X_s$ and $X_{s-1} \xrightarrow{b_s^2} X_s$ for some values of X_s where $b_s^1 \neq b_s^2$.

Finally, we set $\{b_1^1, b_1^2\} \times \{b_2^1, b_2^2\} \times \cdots \times \{b_s^1, b_s^2\}$ is a 2^s multicollision.

4.3 Security Analysis with DMBHT

Theorem 1 *Suppose that a cheating power P^1 on a n-block M is well behaved and that is ϵ admissible, i.e., convincing the answer and ϵ-fraction of verification. Let$E$ = 1/ #Block + $(\partial Y)^l/(Y - C + 1)^C$. Then provided that $(\epsilon - E)$ is positive and non-negligible. It is possible to recover a ρ fraction of the encoded file blocks in $O(Y/\in -\partial)$ interactions.*

Proof depends on Lemma 1.

Lemma 1 *Suppose, a duplicate prover on a n-block file FB is well behaved in the sense above, and that it is $\in$admissible. Let $E = 1/\#Block + (\partial Y)^l/(Y - C + 1)^C$. Then, provided that $(\in -E)$ is positive and non-negligible, it is possible to recover a ∂- fraction of the encoded file blocks in $O(Y/\in -\partial)$ interactions with cheating prover and in $O(Y^2 + (1+\in Y^2)(Y)/(\in -E)$ time overall.*

Proof We say that the extractor's knowledge at each point is a subspace D, represented by a $t \times n$ matrix A in a row reduced level. Suppose that the query response pairs contributing to the extractor's knowledge are $q^1 M = (\mu^1, \ldots \mu_s^1) \ldots q^{(t)} M = (\mu^1, \ldots \mu_s^t)$ or $VM = W$ where V is $t \times n$ matrix whose rows are $\{q^i\}$and w is the $t \times s$matrix whose rows are $((\mu^i, \ldots \mu_s^i))$. The row reduced level matrix A is related to V by $A = UV$ where U is a $t \times t$ matrix with nonzero determinant computed in applying Gaussian elimination to V.

i. The extractor's knowledge initially empty, i.e., $K = 0$.
ii. The extractor's repeats the following behavior until $\#freeK \geq \rho_n$

The extractor chooses random query Q which runs on B. Suppose B chooses to respond giving answer $(\mu, \ldots \mu_s)$ with probability $\in$. Let Q be over indices $I \in [1, n]$ and denoted it in vector notation as q. Now we classify Q into three steps.

i. $q \notin K$
ii. $q \in K$ but $I \not\subset freeK$
iii. $I \subseteq freeK$

For queries, the extractor adds Q to its knowledge K and obtains new knowledge D^1 as follows. It adds a row corresponding to the query V by obtaining V^1 and a row corresponding to the response to W. And obtains W^1 transform matrix U and $U^1$with $A^1 = U^1V^1$. Clearly, a type-1 query increases dimension K by if dimension K equals n then $freeK = n \leq \rho_m$. So the extractor's query phase is guaranteed to terminate by the time. Type-2 quires make up at most a $(1/\#B)$ fraction of the query space. Since

$$\begin{aligned}\Pr\left[\text{Q is type} - 2\right] &= \Pr_Q\left[q \in K \wedge I \not\subset freeK\right]\\ &= \Pr_Q\left[q \in K | I \not\subset K\right]\Pr\left[I \not\subset freeK\right]\\ &\leq \Pr_Q\left[q \in K | I \not\subset freeK\right] \leq (1/\#B)\end{aligned}$$

Theorem 2 *Given a fractional part of the Y-blocks of an encoded file FB, retrieval of complete original file FB with all but negligible probability is possible.*

Proof In our proposed model, we form block tags (FB) at leaf level of DMBHT which are encoded by using rate ϕ erasure code (Tornado-z). As per the properties of Tornado-z code [32] ($k = d + h$), client/user can be capable to retrieve complete file blocks from small fractions (redundant blocks). Tornado-z code with a rate ϕ erasure code is combined in form of original message (d) with redundant data (h), i.e., $r = d + h$. Any client/user can be easily get original message from a sufficient part or number of block packets.

In Fig. 7, we have described a code which is formed by using random equalities (sparse in nature as less number in each equation). Tornado-z code has much faster encoding and decoding than others rate ϕ erasure code. Same as decoding operation, at first, replace received code blocks according to

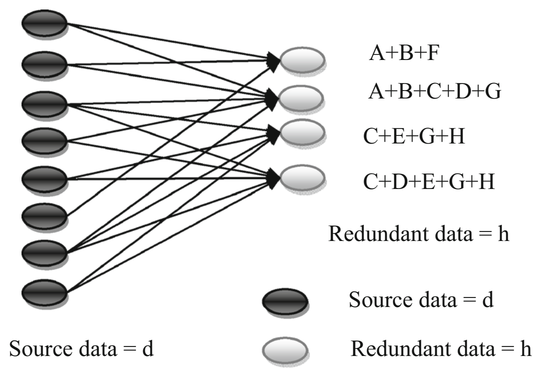

Fig. 7 Description of Tornado-z code

their position in equalities where complete data block values appear. After that apply substitution operation to recover complete blocks and the remaining part of data blocks during recovery at receiver site.

4.4 Security Analysis of Signature Scheme [33]

Theorem 3 *Let us assume signature scheme* $(q, T^!, \in^!)$ *SDH holds in* (G_1, G_2) *then signature model is* $(T, q_s, \in)$ *secure against strong existential forgery under an adaptive chosen message attack given as* $q_s = q$, $\in \geq 2t^! + q_s/p \approx 2 \in^!$ *and* $T \leq T^! - \theta(q^2 t)$ *where* t = *maximum time for exponentiation in* G_1, G_2 *and* Z_p.

Proof using Lemma 2.

Lemma 2 *Suppose signature scheme* $(q, T^!, \in^!)$ *SDH assumption holds in* (G_1, G_2) *then signature model is* $(T, q_s, \in)$*secure against strong existential forgery under an adaptive chosen message attack given as* $q_s = q$, $\in \geq 2t^! + q_s/p \approx 2 \in^!$ *and* $T \leq T^! - \theta(q^2 t)$ *where* t = *maximum time for exponentiation in* G_1, G_2 *and* Z_p.

Proof Let us assume a forger A who will break signature scheme (T, q_s, t). Now, we generate a algorithm B which interacts with A and try to solve q-SDH problem in time $T^!$ with an advantage $\in^!$.

Algorithm B provided a random detail $(g_1, e_1, e_2, \ldots e_q, g_2, F)$of q-SDH problem in (G_1, G_2) where $e_1 = g_1^{x^i} \in G_1$ for $i \in 1, \ldots, q$ and $F = g_2^x \in G_2\{\exists_X X \in Z_p\}$. The objective of algorithm B is to generate a pair of $\left(v, g_1^{1/X+v}\right)$ for $v \in Z_p/\{-v\}$.

4.4.1 Interaction Between Algorithm B and Forger A

Query: Initially, A chooses a collection of message $M_1, M_2, \ldots, M_{q_s} \in Z_p$ $q_s \leq q$. The output of forgerAis q-signed message as initiallyA explains to get of queries Q_r.

Response: On receiving queries from A, simulator B responds along with public parameter and q-signatures on all queries fromA. Let us assume $f(x) = \prod_{i=1}^{q}(x + M_i)$ {polynomial}, further calculate $f(x) = \sum_{i=0}^{q} \beta_i x^i$ where $\beta_0 \beta_1, \ldots, \beta_q \in Z_p$ {coefficients}. Now algorithm B chooses random value $\omega \in Z_p^*$ and compute $g_1^! \rightarrow \prod_{i=0}^{q} D_i^{\beta_i \omega} \in G_1$ and $g_1^! = g_1^{\omega f(x)}$. So by using bilinear properties, algorithm B computes $Z^! = e(g_1^!, g_2)$. At last, the public parameter of A $= \left(g_1^!, g_2, F, Z^!\right)$becomes correct. So $f(x) \neq 0$as all generators of both groups (G_1, G_2) are uniformly distributed for some value of (i) algorithm B may be easily recover secret key (x). In the next phase, simulator B generates group of signatures ρon M_i and form univariate polynomial same as previously defined.

Output: Now the forger A tries to make forgery and compute set of signatures ρ_i^* for forgery and also generates valid set of signatures on$M_* \in Z_p$, $M_* \notin \{M_1 M_2, \ldots, M_q\}$ $\{M_*, \rho_*\}$. So, forger is successful iff $\delta_*/(X + M_*) + \sum_{i=0}^{q-1} \delta_i X^i$ and find $\delta \neq 0$. Now algorithm Bcomputes $\omega-$order root and calculates δ_* mod P.

Finally, the algorithm obtains value $\varpi = \left(g_1^{\delta_*/X+M_*} . g_1^{\sum_{i=0}^{q-1} \delta_i X^i} . \prod_{i=0}^{q-1} g_1^{-\delta_i X_i}\right)^{1/\delta_i} = \left(g^{1/X+M_*}\right)$ So, the algorithm B returns pair of (M_*, ϖ). By this lemma, we prove the weak security toward IND-CPA attack.

5 Performance and Result analysis

In this section, we show the performance and result analysis of our model and compare it with other existing methods. First, we describe about the performance analysis of e-stream encryption/decryption method (ChaCha20) and compare with previously existing AES-CBC 128 bit method. This complete comparison based on computation time for encryption /decryption is shown in Fig. 8.

Further, we also measure the performance analysis of DMBHT with traditional Merkle hash tree (MHT). In Table 1, we have described the comparison between our model with previously existing methods [5,14,17,34,35]. The comparison between traditional MHT and DMBHT is depicted in Fig. 9.

In Table 2, we describe a comparison between rate ϕ erasure codes used for proper retrieval of block tags (FB) at leaf level. The graphical compression of RSC scheme and Tornado-z code scheme in terms of encoding and decoding efficiencies is shown in Figs. 10 and 11.

We have executed our experiments and performed on a system with pairing-based algorithms. The algorithms SSO

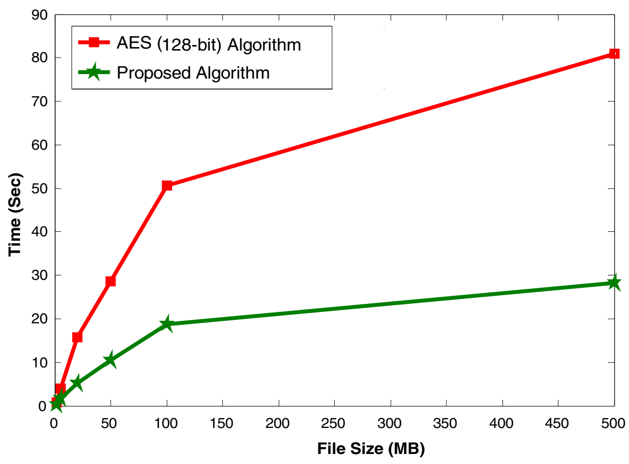

Fig. 8 Comparison between AES-CBC 128 bit and proposed method (ChaCha20)

Table 1 Comparison between BLS signature scheme with SSO scheme on their operation

Scheme	BLS Signature	Short signature without random oracle
Key Stream	1 SM	2 SM
Signing	1 MTP, 1 SM	1 H, 3 PA, 1 INV, 1 SM
Verification	1 MTP, 2 PO	1 H, 3 PA, 1 PO, 1 SM, 1SQ

SM scalar multiplication, *MTP* map to point hash operation in Z_n, *PO* pairing operation, *H* hash function, *INV* inversion in Z_n, *PA* point addition in G_1]

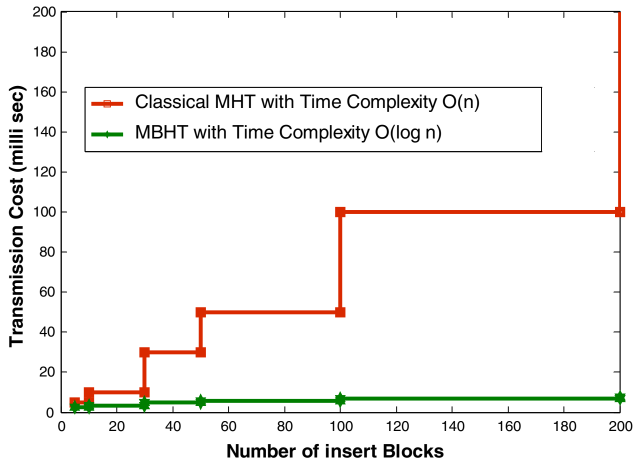

Fig. 9 Comparison between classical MHT and proposed DMBHT

Table 2 Comparison between Reed–Solomon code and Tornado-z code for ($r = d + h$) code

	Tornado-z code	Reed–Solomon code
Decoding inefficiency	$1+\in$	1
Encoding inefficiency	$(d+h)\ln(1/\in)L$	dhl
Decoding time	$(d+h)\ln(1/\in)L$	dhl
Basic operation	XOR	Field operation

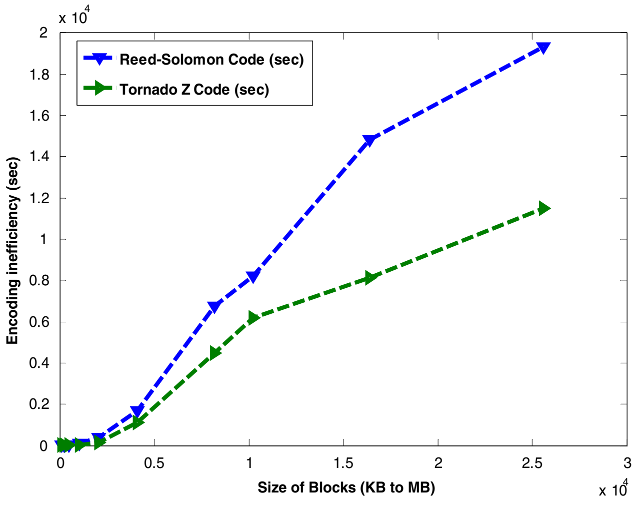

Fig. 10 Graphical compression of encoding inefficiency between RSC scheme and Tornado-z code scheme

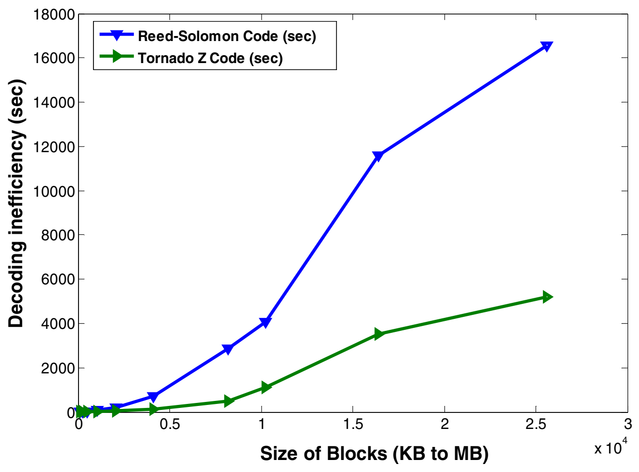

Fig. 11 Graphical compression of decoding between inefficiency of RSC scheme and Tornado-z code scheme

and SHA-512 hash function are implemented by using PBC library version 0.5.14 and crypto library OpenSSL version 1.0.2 to attain 80 bit security token (λ). This exertion is based on 160 bit elliptic curve group on super singular curve over 512 bit finite field. Performance of our method with other existing methodologies is represented in Table 3.

6 Conclusion and Future Scope

In this manuscript, we have addressed some major security concerns related to sensitive data and integrity of VM's disk in virtualization for cloud computing paradigm. In our construction, we have proposed a dynamic secure PoRs scheme that has public auditability, updation, modification, and less time consumption for verification/computation at server and client side with Tornado-z erasure code which has proper retrieval of block tags at leaf level ($k = d + h$) by providing redundant parity data. In the proposed model, we introduced dynamic version of Merkle hash tree with *q-SDH* unforgeable intractable signature scheme to attain proper authentication.

As a future construct, our construction can be further modified by attaining method of constant time with rapid dynamic auditable/updation scheme instead of $O(\log n)$. At the same time, other parameters such as flexibility, reliability, and scalability of proposed method with dynamic Merkle hash B+

Table 3 Different PoRs scheme where $O(\log N)^*$= half computation time than $O(\log N)$

Scheme	BLS	[34]	[14]	[12]	[17]	Our scheme
Data dynamics	Yes	No	No	No	No	Yes
Public auditable	Yes	Yes	Yes	No	No	Yes
Server comp. complexity	$O(c)$	$O(1)$	$O(1)$	$O(1)$	$O(\log N)^*$	$O(\log N)^*$
Verifier comp. complexity	$O(\log N)$	$O(1)$	$O(1)$	$O(1)$	$O(\log N)$	$O(\log N)^*$
Communication complexity	$O(\log N)$	$O(1)$	$O(1)$	$O(1)$	$O(\log N)$	$O(\log N)^*$
Verifier storage complexity	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

tree over AES-CBC will be addressed. This can be further modified by arranging a recovery and backup technique for sensitive data.

Acknowledgements The authors would like to thank anonymous referees and the editor for their valuable suggestions in improving the technical standard of this manuscript. The authors wish to express their gratitude and heartiest thanks to the Department of Computer Science & Engineering, Indian Institute of Technology (ISM), Dhanbad, India, for providing their research support.

References

1. Mell, P.; Grance, T.: The NIST definition of cloud computing. 20–23. (2011)
2. Fox, A.; et al.: Above the Clouds: A Berkeley View of Cloud Computing. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Rep. UCB/EECS. 28:2009 (2009).
3. Barham, P.; et al.: Xen and the art of virtualization. ACM SIGOPS Oper. Syst. Rev. **37.5**, 164–177 (2003)
4. Lombardi, F.; Di Pietro, R.: Secure virtualization for cloud computing. J. Netw. Comput. Appl. **34**(4), 1113–1122 (2011)
5. Ateniese, G.; et al.: Scalable and efficient provable data possession. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks. ACM (2008)
6. Juels, A; Kaliski Jr., B.S.: PORs: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM (2007)
7. Pearson, S.: Taking account of privacy when designing cloud computing services. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society (2009)
8. Gu, L.; Cheung, S.-C.: Constructing and testing privacy-aware services in a cloud computing environment: challenges and opportunities. In: Proceedings of the First Asia-Pacific Symposium on Internetware. ACM (2009)
9. Siebenlist, F.: Challenges and opportunities for virtualized security in the clouds. Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. ACM (2009)
10. Takabi, H.; Joshi, J.B.; Ahn, G.-J.: Security and privacy challenges in cloud computing environments. IEEE Secur Priv **6**, 24–31 (2010)
11. Liang, Q.; Wang, Y.-Z.; Zhang, Y.-H.: Resource virtualization model using hybrid-graph representation and converging algorithm for cloud computing. Int. J. Autom. Comput. **10**(6), 597–606 (2013)
12. Hay, B.; Nance, K.; Bishop, M.: Storm clouds rising: security challenges for IaaS cloud computing. In: 2011 44th Hawaii International Conference on System Sciences (HICSS). IEEE (2011)
13. Zhang, F.; et al.: CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM (2011)
14. Shacham, H.; Waters, B.: Compact proofs of retrievability. In: Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), pp. 90–107 (2008)
15. Wang, Q.; et al.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Computer Security—ESORICS 2009, pp. 355–370. Springer, Berlin (2009)
16. Boneh, D.; Lynn, B.; Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004)
17. Erway, C.Chris; et al.: Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. (TISSEC) **17.4**, 15 (2015)
18. Gemmell, J.: ECRSM—erasure correcting scalable reliable multicast. Vol. 20. Microsoft Research Technical Report MS-TR-97 (1997)
19. Nonnenmacher, J.; Biersack, E.W.; Towsley, D.: Parity-based loss recovery for reliable multicast transmission. IEEE/ACM Trans. Netw. **6**(4), 349–361 (1998)
20. Nonnenmacher, J.; Biersack, E.W.: Asynchronous multicast push: AMP. In: Proceedings of the International Conference on Computer Communication. IOS PRESS (1997).
21. Rizzo, L.; Vicisano, L.: A reliable multicast data distribution protocol based on software FEC techniques. In: Proceedings of The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97), Sani Beach, Chalkidiki, Greece, June (1997)
22. Schooler, E.; Gemmell, J.; Wa, R.: Using multicast FEC to solve the midnight madness problem, vol. 25. Microsoft Research Technical Report MS-TR-97 (1997)
23. Yajnik, M.; Kurose, J.; Towsley, D.: Packet loss correlation in the MBone multicast network. In: In: Global Telecommunications Conference, 1996. GLOBECOM'96.'Communications: The Key to Global Prosperity. IEEE (1996)
24. Bernstein, D.J.: ChaCha: a variant of Salsa20. In: Workshop Record of SASC, vol. 8 (2008)
25. Nir, Y.; Langley, A.: ChaCha20 and Poly1305 for IETF Protocols. RFC 7539. May 2015. http://www.rfc-editor.org/info/rfc7539
26. Zheng, Q.; Xu, S.: Fair and dynamic proofs of retrievability. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy. ACM (2011)
27. Gilbert, H.; Handschuh, H.: Security analysis of SHA-256 and sisters. In: International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, pp. 175–193 (2003)
28. Boneh, D.; Boyen, X.: Short signatures without random oracles. In: Advances in Cryptology—EUROCRYPT 2004. Springer, Berlin (2004)
29. Procter, G.: A security analysis of the composition of ChaCha20 and Poly1305. IACR Cryptol. ePrint Arch. **2014**, 613 (2014)
30. Neve, M.; Tiri, K.: On the complexity of side-channel attacks on AES-256-methodology and quantitative results on cache attacks (2007)
31. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Advances in Cryptology—CRYPTO 2004. Springer, Berlin (2004)

32. Byers, J.W.; et al.: A digital fountain approach to reliable distribution of bulk data. ACM SIGCOMM Comput. Commun. Rev. **28.4**, 56–67 (1998)
33. Zhang, F.; et al.: A new short signature scheme without random oracles from bilinear pairings. IACR Cryptol. ePrint Arch. **2005**, 386 (2005)
34. Ateniese, G.; et al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM (2007)
35. Zhang, F.; Safavi-Naini, R.; Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Public Key Cryptography—PKC, pp. 277–290. Springer, Berlin (2004)