VeriDedup: A Verifiable Cloud Data Deduplication Scheme With Integrity and Duplication Proof

Xixun Yu , Hui Bai, Zheng Yan , Senior Member, IEEE, and Rui Zhang, Member, IEEE

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 20, NO. 1, JANUARY/FEBRUARY 2023

CSP’s Threat Model

- Semi-trusted

Notably, a semi-trusted CSP may modify, tamper or delete the uploaded data driven by some profits. The damage of

Raise three security threats:

1. Snooping the private data of the data holders;
2. Cheating the data holders by providing a wrong duplication check result in order to ask a higher storage fee.
3. Causing data loss due to carelessness of data maintenance.

[14]: it is curious about the contents of stored data, but should perform honestly on data storage in order to gain commercial profits.

Note set

contains several randomized bit sequences

Note generation: On input the hidden function f and the secret keys of a data holder, the data holder outputs a randomized note set S and a position set P according to the uploaded blocks of its file.

Function f generation: On input the security parameter λ, AA outputs a hidden function f which is then applied for note generation.

VeriDedup Construction

- Phase 1: System Setup

DH: DH generates (pk_w, sk_w) for PRK and the key (V_w, s_w) is the Eillptic Curve secret key of data holder.

AA: AA generates a hidden function f and pk_{AA} and sk_{AA} for PRE and broadcast pk_{AA} to the data holders.

CSP:CSP generates a public and secret key pair (e,d),which is used to encode the uploaded tags of the data holders and the CSP for duplication check.

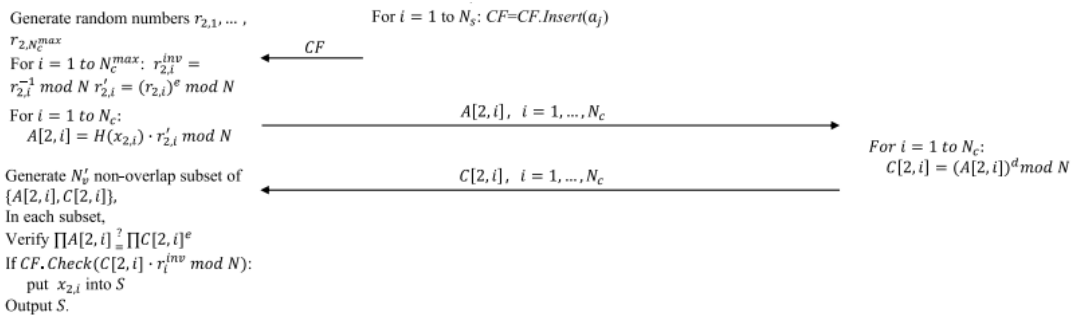
VeriDedup Construction

- Phase 2: Data preprocessing and Duplication Check

Suppose: DH:u_1, u_2; File: f_1, f_2.

Step1:1): Divide f_1 into several splits where each split contains m blocks and adopt ECC to extend d − 1 blocks of redundancy.2) For each block $B_{1,i}$, u_1 generates a block tag $y_{1,i}$. 3) Send the set of tags $\{y_{1,i}\}$to the CSP.

Step2: CSP maintains a set of tags $\{x_j\}$.1)CSP generate a_j=$H(x_j)^d mod\ N$,send $\{a_j$, $H(x_j)$,sign($H(x_j)$)}Δ to AA.2) AA verify, If all the verification passes,AA creates a cuckoo filter CF as input of $\{a_j\}$, i.e., $CF = CF.Insert(\{a_j\})$as a response to u_1 3) as shown in the figure below



VeriDedup Construction

- Phase 3: Note set insertion and Data Upload

Since u_1 is the first to upload a new file that has not been stored by the CSP before. it is served as a data owner and is required to upload its corresponding blocks $\{B_{1,i}\}$.

Step1:

1)u_1 first encrypts $B_{1,i}$ with DKE_1 to get $CT_{1,i}$, which is stored as a X × Y matrix, and encrypts DKE_1 with pk_{AA} to get CK_1;

2) Let $S_{1,i} = \{\eta_{1,i,0}, \ldots \eta_{1,i,k} | f(\eta_{1,i,0}, \ldots \eta_{1,i,k}) = 0\}$be a note set that conforms to the hidden function f;

3) u_1 then inserts all the notes $\{\eta_{1,i,k}\}$ into $CT_{1,i}$ according to the position indexes $P_{1,i}$ to obtain $CTI_{1,i}$ and sends $CTI_{1,i}$ and CK_1 to CSP along with pk_i ;

VeriDedup Construction

- Phase 3: Note set insertion and Data Upload

Step2:

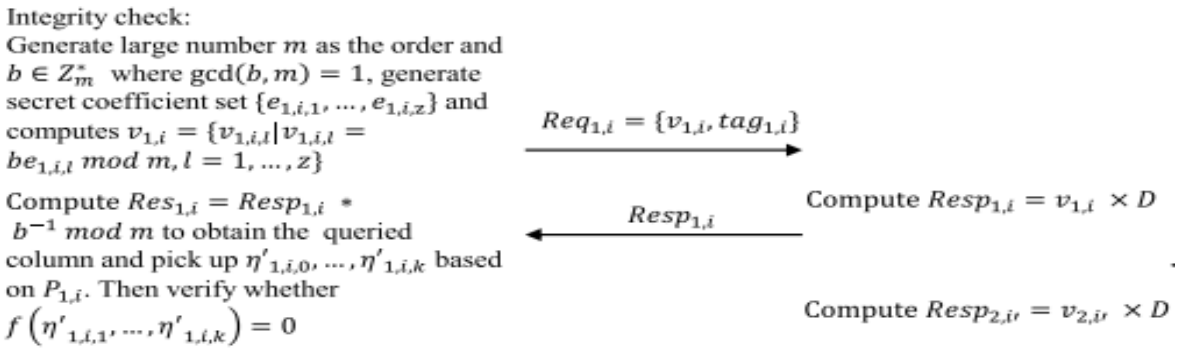
1) u_2 upload the duplication, the CSP first checks the ownership of the blocks. i.e. , $B_{2,i'} = B_{1,i}$.

2) If the verification passes, AA generates re-encryption key $rk_{AA \rightarrow u_j} = RG(pk_{AA}; sk_{AA}; PK_2)$ and send it to CSP.

3) CSP then transfers CK_1 to CK_2 by computing $R\left(rk_{AA \rightarrow u_j}; E(pk_{AA}; DEK_1)\right) = E(pk_2; DEK_1)$ for u2.

At this moment, both u_1 and u_2 can access the same data blocks $B_{1,i}$ $\left(B_{2,i'}\right)$ stored at the CSP and use its corresponding $CT_{1,i}$ $\left(CT_{2,i'}\right)$ to perform the below integrity check.

VeriDedup Construction



- Phase 4: Data integrity check

Suppose: user u_1 challenges the integrity of a single block $B_{1,i}$ stored at CSP.

1. Initializes a large number m and $b \in Z_m^*$, where $\gcd(b,m) = 1$, as a secret.
2. According to position indexes $P_{1,i}$, u_1 then generates a set ($e_{1,i,0}, \dots e_{1,i,z}$) for each column ($x_{1,i,0}, \dots x_{1,i,z}$) with random selected ($d_{1,i,0}, \dots d_{1,i,z}$)

   if $x_{1,i,l} \in P_{1,i}$, $e_{1,i,l} = d_{1,i,l}N^r + N^l$; else $x_{1,i,l} \notin P_{1,i}$, $e_{1,i,l} = d_{1,i,l}N^r$.
3. u_1 computes $v_{1,i} = \{v_{1,i,l} | v_{1,i,l} = be_{1,i,l} mod m, l \in [1, \dots, z]\}$, send $Req_{1,i} = \{ v_{1,i}, tag_{1,i} \}$.
4. CSP computes $Resq_{1,i} = v_{1,i} \times B_{1,i}$ as a response and send it to u_1 .
5. u_1 computes $Res_{1,i} = Resq_{1,i} \times b^{-1} \bmod m$ to obtain the queried columns and then pick out the notes according to the position indexes $P_{1,i}$ to check whether the notes are conformed to the hidden function f.

VeriDedup Construction

- Phase 5: Data Download

u_1 wants to download F_1.

1. Send a request and file name to CSP
2. CSP checks if u_1 has the authorization to download the file , If passed, CSP returns the corresponding block sets $\{CTI_{1,i}\}$ to u1
3. u_1 extracts all the notes according to the position indexes $P_{1,i}$ on each block to get the ciphertexts $CT_{1,i}$ and decrypts each $CT_{1,i}$ using DEK_1 directly to obtain $B_{1,i}$

Performance Evaluation

theoretical analysis on integrity check performance

- Split a 4 GB file into a number of blocks with 128 KB, so that we got each block containing 16384 elements and each element is 64 bits.
- Selected the hidden function as $note[1]||note[2] = (Hash_{SHA256}(note[3]||note[4]))_{128}$.
- Inserted 8 pairs of notes as verification tags into each block, The size of each notes is 64bits.

TABLE 3
Computational Complexity and Communication Cost of TDICP Compared With Existing Works

Scheme	Parameter	Setup cost	Storage overhead	Server cost	Verifier cost	Communication cost
[7]	Block size: 2 KB tag size: 128 B	4.4×10^6 exp 2.2×10^6 mul	tags: 267 MB	764 PRP 764 PRF 765 exp 1528 mul	Challenge: 1 exp Verfi: 766 exp 764 PRP	Challenge: 168 B Response: 148 B
[15]	Block size: 128 bits Number of sentinels: 2×10^6	2×10^6 PRF	sentinels: 30.6 MB	$\perp$	Challenge: 1719 PRF Verfi: $\perp$	Challenge: 6 KB Response: 26.9 MB
[18]	Block size: 80 bits Number of blocks: in one split: 160 tag size: 80 bits	1 enc 5.4×10^6 PRF 1.1×10^9 mul	tags: 51 MB	7245 mul	Challenge: 1 enc 1 MAC Verfi: 45 PRF 160+205 mul	Challenge: 1.9 KB Response: 1.6 KB
[16]	Block size: 160 bits Number of blocks in one split: 160	2.2×10^8 mul 1.4×10^6 PRF	tags: 26 MB	160 exp 2.6×10^5 mul	Challenge: $\perp$ Verfi: 2 exp 1639 PRF 1639 mul	Challenge: 36 KB Response: 60 B
[20]	Block size: 256 bits Number of blocks in one split: 4096	2.6×10^5 PRF 2.6×10^5 PRP	Watchdogs: 8 MB	6.2×10^8 mul	Challenge: 2.0×10^6 mul Verfi: 1.4×10^5 mul	Challenge: 23.3 MB Response: 26.2 MB
Ours	Element size: 64 bits Number of elements in one block: 16384	1.0×10^6 PRP 7.9×10^5 PRF 2.6×10^5 HASH	Positions of hidden parameters: 1.875 MB	3.2×10^7 mul	Challenge: 4.6×10^5 mul Verif: worst: 2.5×10^5 mul 1.7×10^3 HASH best: 2.4×10^5 mul 1.7×10^3 HASH	Challenge: 9.24 MB Response: 9.31 MB

exp: exponentiation; mul: multiplication; PRP: pseudo-random permutation; PRF: pseudo-random function; enc: encryption; MAC: message authentication code.

	Parameter	**Setup cost**	**Storage overhead**	**Server cost**	**Verifier cost**	**Comm. cost**
[3]	block size: 2 KB tag size: 128 B	4.4×10^6 exp 2.2×10^6 mul	tags: 267 MB	764 PRP 764 PRF 765 exp 1528 mul	challenge: 1 exp verif: 766 exp 764 PRP	challenge: 168 B response: 148 B
[1]	block size: 128 bits number of sentinels: 2×10^6	2×10^6 PRF	sentinels: 30.6 MB	$\perp$	challenge: 1719 PRP verif: $\perp$	challenge: 6 KB response: 26.9 MB
[2]	block size: 80 bits number of blocks in one split: 160 tag size: 80 bits	1 enc 5.4×10^6 PRF 1.1×10^9 mul	tags: 51 MB	7245 mul	challenge: 1 enc, 1 MAC verif: 45 PRF, 160 + 205 mul	challenge: 1.9 KB response: 1.6 KB
[4]	block size: 160 bits number of blocks in one split: 160	2.2×10^8 mul 1.4×10^6 PRF	tags: 26 MB	160 exp $2.6 * 10^5$ mul	challenge: $\perp$ verif: 2 exp, 1639 PRF, 1639 mul	challenge: 36 KB response: 60 B
SG	block size: 256 bits number of blocks in one split: 4096	2.6×10^5 PRF 2.6×10^5 PRP	watchdogs: 8 MB	6.2×10^8 mul	challenge: 2.0×10^6 mul verif: 1.4×10^5 mul	challenge: 23.3 MB response: 26.2 MB

Table 2: Comparison of relevant related work with **StealthGuard (SG)**.

VeriDedup的性能评估直接采用的是StealthGuard的评估指标和数据

Performance Evaluation

theoretically analyze the performance of duplication check

Analyze the computational complexity and communication cost of each entity

- Computational complexity of data holder at preprocessing phase
- Computational complexity of CSP at filter generation and duplication check phase
- Computational complexity of AA at filter generation phase
- Computational complexity of data holder at duplication check phase
- Communication cost from CSP to AA
- Communication cost between data holder and CSP

TABLE 4
Computational Complexity of UDDCP

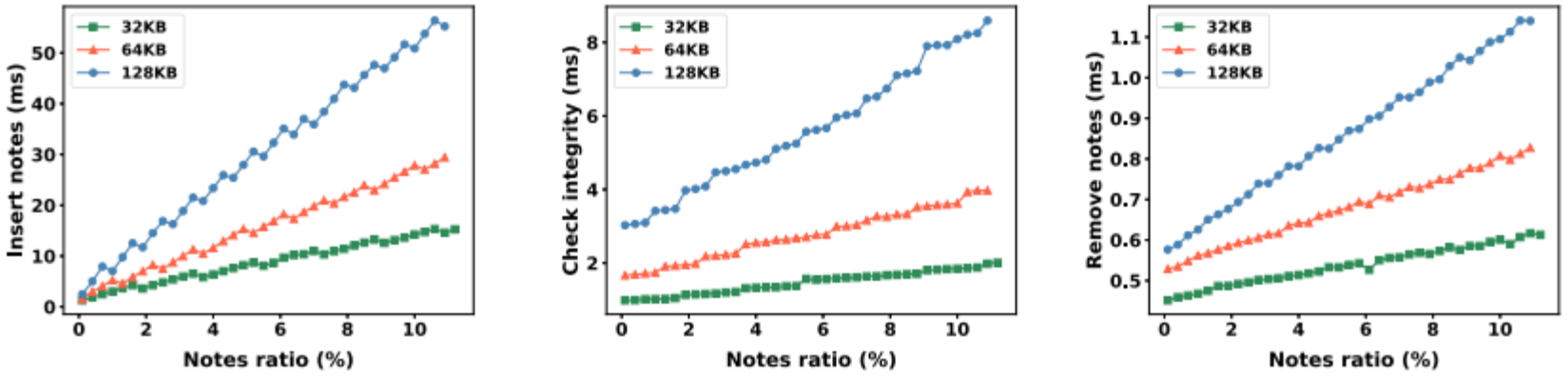
	Initialization	Filter generation	Duplication check
Data holder	N_c PRF		
	N_c INV		$2N_c$ mul
	N_c exp		N_c exp
CSP		N_s exp	N_c exp
AA		$N_s + N_s/\lambda$ exp	
		$2N_s$ mul	
		N_s CF.insert	

exp: exponentiation; mul: multiplication; PRF: pseudo-random function; INV: inversion.

Performance Evaluation

performance evaluation result of TDICP

- we propose the novel note set as the verification tags, the note ratio



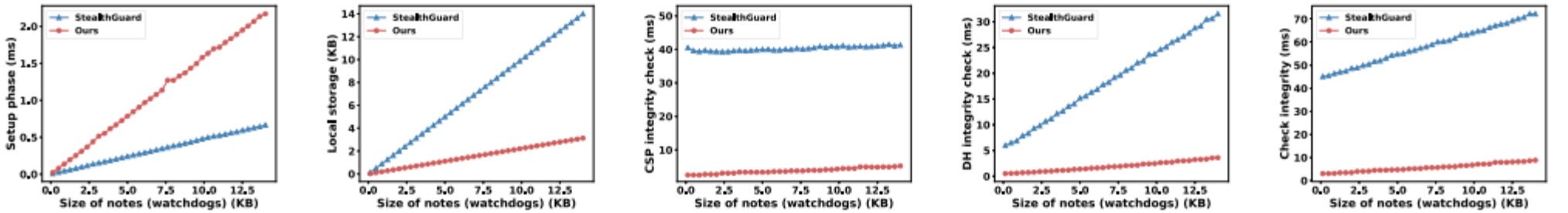
(a) Inserting note cost (b) Integrity check cost (c) Removing note cost

Fig. 5. Computational costs with regard to note ratio varying from 0.02 to 0.10.

Performance Evaluation

performance evaluation result of TDICP

- Compare it with StealthGuard in terms of setup cost,integrity check cost at the CSP and the data holder (DH).



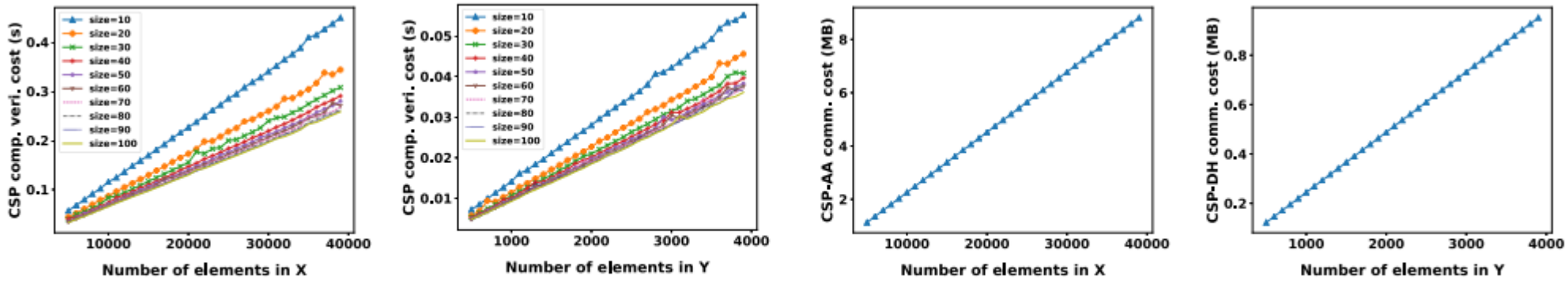
(a) DH Setup cost (b) DH storage overhead (c) CSP integrity check cost (d) DH integrity check cost (e) Total integrity check cost

Fig. 6. Computational costs with regard to note size varying from 2KB to 14KB compared with StealthGuard.

Performance Evaluation

performance evaluation result of UDDCP

focuses on the AA and data holder verification on the CSP computations and evaluated UDDCP performance in various sizes of the nonoverlap subset as we introduce batch verification into UDDCP



(a) Veri. cost of AA (b) Veri. cost of DH (c) CSP-AA comm. cost (d) CSP-DH comm. cost

Fig. 7. Computational costs with regard to the size of the non-overlap subset.

- 论文并没有明确提出CSP的semi-trusted 的可信在什么地方，该论文是基于论文[14]进行改进的，在论文[14]中说明trust为诚实的存储数据。

- 论文中对位置索引集$P_{1,i}$ 的选取貌似没有说明，好像就是将数据块行列打乱然后用户选择。且并没有对其进行安全保护，若获得用户的位置索引集，应该可以提取出插入到文件块中的note set。

- 论文中对完整性检测的性能评估主要在各个实体的计算开销和通信开销上。其评估指标和对比对象是在论文[20]StealthGuard的性能对比情况上加上了自己的数据。