INFOCOM 2022 1570747434

On Designing Secure Cross-user Redundancy Elimination for WAN Optimization

Yuan Zhang, Ziwei Zhang‡, Minze Xu†, Chen Tian, Sheng Zhong
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
Emails: {zhangyuan, tianchen, zhongsheng}@nju.edu.cn, cnmzxu@gmail.com†, meilizilanwei@163.com‡

***Abstract*—Redundancy elimination (RE) systems allow network users to remove duplicate parts in their messages by introducing caches at both message senders' and receivers' sides. While RE systems have been successfully deployed for handling unencrypted traffic, making them work over encrypted links is still open. A few solutions have been proposed recently, however they either completely violate end-to-end security or focus on single-user setting. In this paper, we present a highly secure RE solution which supports cross-user redundancy eliminations on encrypted traffics. Our solution not only preserves the end-to-end security against outside adversaries, but also protects users' privacy against semi-honest RE agents. Furthermore, our solution can defend malicious users' poisoning attack, which is crucial for cross-user RE systems but has never been studied before. In cross-user RE systems, since all users inside a LAN write into a shared, global cache and use it to recover their original messages from deduplicated ones, the poisoning attack is prone to happen, and cause systematic damage to all users even when only one user is malicious and injects poisoned data into the cache. We rigorously prove our solution's security properties, and demonstrate its promising performance via testing the proof-of-concept implementation with real-world internet traffic data.**

***Index Terms*—redundancy elimination, WAN optimizer, Message-locked encryption, poisoning attack**

I. Introduction

Several studies (e.g. [1]–[4]) have discovered that a large amount of same or similar contents are transferred repeatedly across the Internet. To reduce the redundancy in network traffic, especially in the wide-area network (WAN) traffic, a number of redundancy elimination (RE) systems or protocols [5]–[7] have been proposed by researchers from both the industry and the academia in recent years.

A RE system for WAN optimization involves two or more LANs which are connected with WAN links. On an edge node of each LAN (usually a gateway router), a module called deduplication *agent* is deployed to serve all RE system *users* inside the LAN. The agent maintains a cache which stores inter-LAN messages that are recently sent or received by users inside the LAN. Before a new message enters the WAN, its sender cooperates with the agent to eliminate the redundant parts.

Specifically, to increase the chance of deduplication, messages are generally broken into chunks. The cache serves as a "chunk dictionary", and stores chunks and their identifiers or *fingerprints*. Before a sender sends a message cross the WAN, it first looks up the message' chunks in the cache and replaces matched ones with shorter identifiers or *fingerprints* which are generally the chunks' hash values. Thus the total amount of data that traverses through the WAN is reduced. Eliminating redundant or repeating data of WAN traffic can largely increase the network's efficiency and reduce the WAN cost, thus RE systems are of great interest to enterprises (especially big ones who have branches in different cities or even countries), ISPs, data centers and networking device vendors.

In this paper, we aim to design secure RE solutions that support cross-user deduplications on encrypted traffic. Although RE systems have been successfully deployed to handle unencrypted traffic, making them work on encrypted traffic and support cross-user deduplications turns out to be extremely challenging. Especially existing secure transmission protocols such as TLS and IPSec adopt so-called "End-to-end encryptions" which requires that messages are encrypted by their sender before entering into the network environment, and gets decrypted only at the receiver side after leaving the network. Deduplication agents who reside in the middle can only see garbled meaningless bits in this case, thus cannot find the matching parts easily as they do for unencrypted traffic. In addition, compared with single-sender-single-receiver RE scenarios, multi-sender-multi-receiver or cross-user deduplications make deduplication agents' task even more complicated since deduplications need to be performed over messages that are encrypted by different users possibly using different keys.

Besides above difficulties, we further consider users may be malicious and launch the *poisoning attack*. In a cross-user RE system, a shared, global cache is used by the agent and users inside the same LAN to perform deduplications. A malicious user can launch this lethal attack by injecting poisoned or false content into the cache and trick other users to receive false messages when they use the poisoned content in the cache to recover deduplicated messages. Given that LANs usually have many users with varied networking security conditions and the cache is fed by all users, poisoning attack is prone to happen

This work was supported in part by the National Natural Science Foundation of China under Grant Numbers NSFC-61872179, NSFC-62072228, NSFC-61902176, BK20190294, NSFC-61872176, and the Leading-edge Technology Program of Jiangsu NSF (BK20202001). Minze Xu and Sheng Zhong are the corresponding authors.

and can cause system-wide damages to all users inside the RE system even if one user is compromised or controlled by malicious attackers.

One may propose to let the sender attach an error detection code to the message so that the receiver can detect the poisoned message and demand a re-transmission. However, this simple fix is not a cure to poisoning attacks. The poisoned content is still inside the cache and continues to cause damage (i.e. extra bandwidth cost, delay due to retransmissions) to all users unless the poisoned content is removed completely.

Moreover, we note that letting the agent remove a poisoned record from its cache is more tricky than it appears to be. Firstly, the agent needs to be able to verify the correctness of users' reports of poisoned records, otherwise reporting mechanism could be misused by malicious users. Secondly, for privacy matters, users need to convince the agent that a stored ciphertext of a chunk is poisoned without revealing the chunk to the agent.

We address the above complicated issues, and design a fully secure cross-user RE protocol called DTEp that preserves end-to-end security and thwart poisoning attacks in this paper.

To achieve the end-to-end security, DTEp adopts a different "deduplication-then-encrypt" strategy. Specifically, DTEp lets users and their agent to jointly maintain a two-level cache system, and perform *D*eduplication first (according to the cache) and *T*hen *E*ncryptions over reduced messages using TLS or SSL. Instead of storing previously sent message chunks, the cache owned by the agent stores the ciphertexts of these chucks for protecting the confidentiality of messages.

To support secure cross-user deduplications and protect the system against poisoning attacks simultaneously, we propose a novel Message-Locked Encryption scheme called MLEvd using prime-order bilinear maps for message encryption, and adopt a lightweight "key revealing" approach for (indirect) ciphertext verification instead of using costly zero-knowledge proofs based solutions. MLEvd allows different users to generate same keys for same plaintexts so that cross-user deduplication is possible, and additionally supports key verification and dynamic keys. Accordingly, users can report poisoned ciphertexts to the agent and prove they are indeed poisoned by revealing the encryption key to agent. With the help of MLEvd's key verification, the agent can easily verify the correctness of the key first, and then the ciphertext by performing the encryption with the correct key on its own. Since MLEvd supports dynamic keys, users can generate a new key to encrypt once a key has been revealed.

Our main contributions are summarized as follows.

- We study the cross-user RE system for WAN optimization over encrypted traffic and propose the first fully secure RE protocol called DTEp that preserves end-to-end security and thwarts poisoning attacks. We formally prove the security of our protocol.
- We propose a novel MLE scheme called MLEvd that supports dynamic key and key verification. MLEvd is PRV-CDA secure, i.e. semantically secure under chosen distribution attacks, which is the best possible type of privacy for MLE schemes [8]. It guarantees no PPT adversaries can differentiate the encryptions of two equal-length messages that are unpredictable or have high min-entropy.
- We implement a proof-of-concept of our protocol, and use real-world data set to evaluate the performance of it.

The rest of this paper is organized as follows. In Sec. II and Sec. III, we introduce the related works and preliminaries respectively. Then, we explain MLEvd in Sec. V and present DTEp in Sec. VI. After showcasing DTEp's performance in Sec. VII, we conclude our paper in Sec. VIII .

II. Related work

A. WAN Redundancy elimination

Previous RE systems are usually designed for reducing unencrypted WAN traffic [7]. Due to the rapid increase of encrypted traffic of Internet, researchers began to design RE systems for encrypted traffics.

In [5], Cisco Inc. designs a RE system over SSL encrypted links in its Wide Area Application Services SSL Application Optimizer. Cisco's RE system lets the agent to launch a man-in-the-middle attack to decrypt the traffic encrypted by the sender. Since the agent can see the decrypted message, it can perform deduplications easily based on plaintext. Although the system protects privacy against outside adversaries, the system actually breaks the end-to-end privacy between the sender and the receiver and reveals the message to the agent.

In [6], Fan et. al. propose a novel two-layer encryption RE scheme called REET for WAN traffic deduplications. To largely preserve the end-to-end privacy and support the agent-side deduplication (meaning the agent performs deduplications over the ciphertexts that are encrypted by users), REET relies on two layers of encryptions. It first breaks a message into several segments which consists of one "fingerprint chunk" and a few "payload chunks". Then it lets users firstly encrypt the all chunks using a deterministic symmetric encryption scheme, and then re-encrypt the first-layer ciphertexts of payload chunks using a randomized symmetric encryption scheme with their fingerprint chunk's hash as the key. Finally, users need to also run a secret sharing scheme on the key and attach the random share to the ciphertext. When the agent sees two ciphertexts with the same fingerprints, the agent also gets two shares of the second-layer key. By recovering the key, the agent can remove the second-layer encryption of the payload chunks and see whether redundancy can be found. This work improves the security of RE systems since the agent needs to first see a hint of possible redundancy (i.e. the same fingerprint) then can be able to check the other chunks. However, two layers of encryptions bring extra burden to the users and authors do not consider poisoning attacks.

B. Convergent encryptions and Message-locked encryptions

Nowadays, encryption schemes usually are required to be *semantically secure* which basically means no adversary can extract any information about the plaintexts from their corresponding ciphertexts. This immediately renders deduplications

over the ciphertexts impossible since we need the server to tell whether two ciphertexts have the same plaintext.

To meet the practical need, a less-secure encryption scheme called convergent encryption (CE) was first proposed by Douceur et. al. [9]. For a message m, CE uses the hash value of m as the encryption key and use a deterministic symmetric encryption to encrypt the message. It is easy to see the same messages result in the same ciphertexts, which makes the agent be able to perform deduplications over ciphertexts. Despite CE schemes had been widely used to construct secure deduplication solutions, most work failed to provide rigorous treating on the security of CE schemes until Xu et. al. [10] and Bellare et. al. [8] present their excellent jobs. Specifically, CE was generalized into a type of encryption schemes called Message-Locked Encryptions, and semantically security models under "chosen distribution attacks" named PRV-CDA, PRV$-CDA and their adaptive versions are defined assuming the underlying message is unpredictable [8]. In this paper, we adopt the same theoretical framework and design a novel MLE scheme that is proved to be PRV-CDA secure.

C. Storage deduplication and Tag consistency

The similar idea of deduplication or redundancy elimination is also studied for online storage systems such as cloud or IoT, and related studies (e.g. [8], [10]–[14]) are quite popular in recent years.

Despite the similarity, deduplication problems in the two areas have many major differences. Most secure deduplication works for storage systems work on the file-level, so chunking is generally not required. In addition, the ownership of the stored files is a major concern for storage RE systems while the concept usually does not exist in networking systems. Finally, it is important for a network RE solution to provide high throughput. However, this is usually not a concern for storage RE solutions.

Interestingly, we point out the poisoning attack (or the equivalent one named "duplicate faking" [8]) has been found on an existing RE storage solutions [15], and have been discussed in a few works [8], [10], [16]. Solutions to this problem in storage systems require the RE systems to have a security property called "tag consistency".

Tag consistency can be achieved by requiring the server to be able to identify a false ciphertext at the moment when it is uploaded. It is generally implemented in two manners. Some works make the tag to be directly verified with the ciphertext. For example, the tag is computed as the hash value of the ciphertext in CE. Other works achieve tag consistency by letting the user upload a zero-knowledge proof which can be used by the server to verify whether the ciphertext and the tag match. However, the first kind of solutions only works with deterministic ciphertexts thus have weak security and the second kind of solutions incurs heavy cost in our problems.

Another way to provide tag consistency is by allowing the user to perform a tag recompute-and-check operation (called "guarded decryption" in [8]). However, as we have explained before, guarded decryption only lets the user observe the wrong ciphertext, but it still needs to convince the agent about the attack in our problem.

III. Preliminaries

A. System model

To provide a simple motivating scenario, we consider an enterprise which has two branches in two different cities as depicted in Fig. 1. Each branch has its own LAN, and the two LANs are connected with a public WAN. We assume a deduplication *agent* resides at the gateway node in each LAN and helps the two LANs' *user* nodes to perform deduplications over their WAN communication traffic. For security considerations, users communicate with each other via TLS connections which offers the applauding end-to-end security.

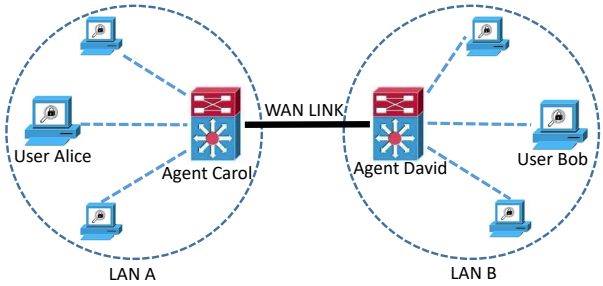

Fig. 1. System Overview

To perform deduplications over the encrypted connections, users and agents need to cooperate and run a redundancy elimination protocol in a secure manner.

B. Adversary models

We consider two kinds of adversaries in our RE system:

1) Semi-honest agents: We assume the deduplication agents are *semi-honest*. In essence, this requires agents to honestly follow the predefined protocol. Nevertheless, the agent may try to infer private data of other protocol participants (i.e. the users) based on the data collected during the protocol execution.
2) Malicious users: Due to accidents such as hacking, software bugs, etc., users could be malicious and violate the protocol. Specifically, we focus on the poisoning attacks, i.e. injecting incorrect ciphertexts into the RE system from these users.

We note that there might be another kind of adversary called *outside eavesdropper*. In our scenario, an eavesdropper is not a LAN user, but takes the advantage of the open, public WAN to monitor the data traffic between the two LANs and to break the communication confidentiality. Since all WAN data traffic is encrypted via TLS in our design, eavesdropping attacks can be effectively thwarted. We neglect this adversary in our paper.

C. Content-defined chunking

To increase the chance of finding redundancy, messages or files to be deduplicated are generally cut into smaller chunks using content-defined chunking (CDC) schemes. One of the

most fundamental and popular CDC schemes is the Rabin CDC algorithm which is also adopted in our experiments.

Specifically, given a message, Rabin CDC algorithm would run a sliding window through the bits of the message, and compute the Rabin fingerprint [17], [18] of the content inside the window. Whenever the fingerprint satisfies a preset condition is satisfied (e.g. a multiple of 2^k), the message is cut at the current window's boundary. The above scanning and cut process ends until the window reaches the last bit of the message.

We note that works (e.g. [19], [20]) have been proposed in recent years to increase the performance of CDC schemes for deduplication systems.

D. Bilinear maps

A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is a function that maps a pair of elements in two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to an element in the third group $\mathbb{G}_3$, and satisfies the following property:

- *bilinear*: $e(u^a, v^b) = e(u, v)^{ab}$ holds for all $a, b \in \mathbb{Z}$, $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$.

Usually we further require e is *non-degenerate*, i.e. e does not map every pair to the identity.

Specifically, our protocol uses a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}^\star$ in which Computational Diffie-Hellman (CDH) problem is hard on $\mathbb{G}$, and let g and $g^\star$ be the two groups' generators respectively.

In our solution, we further require the order the groups is a large prime.

IV. DTE-s: A STRAWMAN PROPOSAL

For clarity, we first propose DTE-s, which outlines the main structure of our DTE-p solution, but adopts a naive message-lock encryption scheme MLE := (MLE_KG, MLE_ENC, MLE_DEC) and totally neglects the poisoning attacks.

A. Protocol details

1) Chunking: For each message M to be sent, Alice runs a CDC algorithm and decomposes M into an array of small chunks:

$$M = \{m_1, m_2, \ldots, m_n\} \tag{1}$$

2) Key and fingerprint generation: For each chunk m_i ($i \in \{1, 2, \ldots, n\}$), Alice computes its key as

$$K_i = \text{MLE\_KG}(m_i) := h(m_i), \tag{2}$$

and fingerprint as

$$F_i = h(K_i), \tag{3}$$

where h is a publicly agreed secure hash function.

3) Deduplication: For each chuck, Alice checks whether it can be deduplicated by looking up its corresponding fingerprint in the fingerprint table T_{La} stored in her local cache. If the fingerprint is found, the chunk has been sent and Alice can send a short key instead of the long chunk. With the key, the receiver can find the right ciphertext and decrypt the chunk out.

After deduplication, Alice gets a deduplicated message $M' = \{m'_1, m'_2, \ldots, m'_n\}$ as follows.

$$m'_i = \begin{cases} m_i & \text{if } F_i \notin T_{La}; \\ K_i & \text{otherwise.} \end{cases} \tag{4}$$

4) End-to-end secure transmission: Alice establishes a TLS connection with Bob, and sends M' to Bob through the connection. The end-to-end security of TLS ensures only Bob can see M'.

5) Message recovery: After receiving a TLS encrypted message, Bob decrypts it and gets M'. Bob recovers M from M' with the help of David, the deduplication agent in his LAN.

Specifically, for each K_i in M' Bob computes its corresponding fingerprint F_i following Eq. (3),[1] and downloads its corresponding ciphertext c_i from David by sending F_i to him. After receiving c_i, Bob decrypts it to recover

$$m_i = \text{MLE\_DEC}(c_i, K_i). \tag{5}$$

and replaces K_i with m_i in M' to recover M.

6) Synchronizations of Alice: For each $F_i \notin T_{La}$, Alice encrypts m_i with K_i as

$$c_i = \text{MLE\_ENC}(m_i, K_i). \tag{6}$$

Alice sends all $< F_i, c_i >$ to its deduplication agent Carol. She downloads all fingerprints in the global fingerprint table T_{Gc} from Carol's cache and inserts them to T_{La} in her local cache.

7) Synchronizations of Carol: After receiving a list of $< F_i, c_i >$, Carol inserts them to T_{Gc}, and sends back the list of fingerprints in T_{Gc} to Alice.

8) Synchronizations of Bob: For each M_i received, Bob computes K_i following Eq.(2) and encrypts m_i using it as the key

$$c_i = \text{MLE\_ENC}(m_i, K_i). \tag{7}$$

In addition, Bob computes F_i following Eq.(3).

Bob uploads all $< F_i, c_i >$ pairs to David.

9) Synchronizations of David: After receiving a list of $< F_i, c_i >$ from Bob, David inserts them to his global fingerprint table T_{Gd} and sends back the latest list of fingerprints in T_{Gd} to Bob.

[1]Here we assume everyone agrees on a coding scheme that differentiates message chunks and keys in a deduplicated message.

B. A few notes on DTEs

DTEs adopts a two-level cache system. Each user maintains a local cache and the agent maintains a global cache. To increase the efficiency of message transmission, users can check the redundancies locally. This is important for real-world networking systems.

In addition, we note the synchronizations between the user and the agent can take place during their idle time in practice. There might be temporary inconsistencies between the user's local cache and the agent's global cache. However, the damage caused by the inconsistency could only be a lower deduplication efficiency, rather than a transmission failure since one can introduce an error detection mechanism (e.g the error detection code) and data re-transmission mechanism in case Bob fails to decrypt the correct chunk.

Finally, we note that since DTEs only adopts a naive MLE scheme, it cannot achieve the formal PRV-CDA security guarantee, and fails to protect the system against poisoning attacks as we will clarify in the next section.

V. MLEvd: a MLE scheme supporting key verifications and dynamic keys

Before we introduce our DTEp protocol, we propose a novel MLE scheme which would be used to construct DTEp.

A. The tricky poisoning attack

We note that it is easy for the message receiver Bob to detect any poisoning attack assuming an error detection code is attached within the message (which is quite common in many application-layer protocols) and to locate the poisoned ciphertexts if they exist. Specifically, if Bob finds the recovered message and the error detection code do not match, Bob knows some ciphertexts are poisoned. In addition, Bob can easily check whether a ciphertext $\tilde{c}$ is poisoned by computing the key himself with the recovered plaintext $\tilde{m}$:

$$\tilde{m} = \text{MLE\_DEC}(\tilde{c}, K), \tag{8}$$

$$\tilde{K} = \text{MLE\_KG}(\tilde{m}). \tag{9}$$

If $\tilde{K} \neq K$, Bob knows $\tilde{c}$ is poisoned. Once Bob knows which ciphertexts are poisoned, he can send these ciphertexts' fingerprints back to Alice and require Alice to re-send the corresponding chunks.[2]

To completely remove a poisoned ciphertext, Bob still needs to notify David of this ciphertext and convince him that it is poisoned since a malicious inside attacker may deliberately send a fake alarm.

Unfortunately, it is challenging for Bob to convince David that a fingerprint-ciphertext pair $\langle F_i, \tilde{c}_i \rangle$ is poisoned, or not the correct encryption of m_i. A naive solution is to reveal m_i to David, and David can verify $\tilde{c}_i$'s correctness as Bob does the verification. However, this completely loses Bob's message secrecy, thus is not acceptable. One might propose to use zero-knowledge proofs (ZKPs) which are used by Cryptographers to prove a statement about a secret without revealing the secret. However, symmetric encryption schemes (e.g. AES, ChaCha, etc.) are mostly adopted to achieve high throughput in practical network traffic transmissions. Therefore, Bob needs to prove a ciphertext is or is not the correct AES/ChaCha encryption of a secret input essentially. The encryptions here are not based on public-key assumptions and do not have simple arithmetic structure which render the ZKP solutions costly in terms of computation and communication.

[2]We assume that Alice maintains a buffer which stores recent sent chunks and their fingerprints.

B. Details of MLEvd

To conquer the above challenges, we design MLEvd which supports key verifications and dynamic keys as follows.

MLEvd adopts a dynamic key $K_i(t)$ generated based on the plaintext chunk m_i and a random nonce t. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}^\star$ be a non-degenerate bilinear map such that the CDH assumption holds on $\mathbb{G}$. Let SDE $=$ (SK,SE,SD) be a symmetric deterministic encryption scheme where SK, SE and SD are the key generation, encryption, decryption algorithms respectively. And denote by $\mathcal{K}$ the key space of SDE.

Specifically, MLEvd consists of six PT algorithms:

- PG$(1^\lambda) \to P$: on input of the security parameter 1^λ, *parameter generation* algorithm generates pairing group $\mathbb{G} = \langle g \rangle$ of a prime order p, its corresponding bilinear map e and the target group $\mathbb{G}^\star$. Let $h : \{0,1\}^* \to \mathbb{Z}_p^*$ and $H : \{0,1\}^* \to \mathcal{K}$ be two cryptographic hash functions. Then KG returns $\{e, g, g^*, p, h, H\}$ as the public parameters P.
- FG$(P, m) \to F$: on input m, the *fingerprint* algorithm computes its fingerprint as
$$F = g^{h(m)}. \tag{10}$$
- KG$(P, m) \to (K(t), t)$: on input input m, the *key generation* algorithm samples a random nonce $t \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$ and computes
$$K(t) = g^{h(m)t}, \tag{11}$$
and outputs $(K(t), t)$.
- ENC$(P, m) \to C$: on input m, the *encryption* algorithm calls KG to get $(K(t), t)$, computes
$$c = \text{SE}(m, H(K(t))), \tag{12}$$
$$T = g^t, \tag{13}$$
and outputs $C = (c, T)$ as the ciphertext.
- DEC$(C, K(t)) \to m'$: the *decryption* algorithm parses c and T from C, and outputs
$$m' = \text{SD}(c, H(K(t))). \tag{14}$$
The *guarded decryption* algorithm GDEC$(C, K(t)) \to \{m', \perp\}$ can be further defined by additionally checking whether
$$T^{h(m')} = K(t) \tag{15}$$
holds, and outputting $\perp$ instead when result is negative.

- $\mathrm{KV}(C, F, K) \rightarrow \{0, 1\}$: the *key verification* algorithm parses c and T from C, checks whether

$$e(F, T) = e(g, K) \tag{16}$$

holds. KV outputs 1 if yes, and 0 otherwise.

We note that guarded decryption algorithm can be used by users to check if a ciphertext is poisoned. Using the correct key, the algorithm always outputs the correct plaintext if the ciphertext is right, and outputs a false symbol $\perp$ if a poisoned or incorrect ciphertext is inputted.

In addition, the key verification algorithm can be used by agents to verify if a key submitted by users (who want to prove a poisoned attack to agents) is the correct one corresponding to the ciphertext and fingerprint. If the result is positive, the algorithm outputs 1; otherwise, it outputs 0.

C. Correctness analysis

We postpone the security analysis till we finish introducing the entire protocol. Here we briefly analyze the correctness of MLEvd. Since correctness for regular encryption and decryption directly follows the correctness of underlying encryption scheme SDE, we neglect this part and focus on proving the correctness of guarded decryption and key verification algorithms. Specifically, we have proved the following theorems.

Theorem 1. *The guarded decryption of MLEvd is **correct** in the sense that with an input of a correctly computed key $K(t) = T^{h(m)}$, GDEC$(C, K(t))$ outputs m if C correctly encrypts m, or $\perp$ otherwise.*

Proof. Considering C is correctly computed from m following Eq. (12) and Eq. (13), the correctness of underlying SDE guarantees $m' = m$. Then we have

$$T^{h(m')} = T^{h(m)} \tag{17}$$

$$= (g^t)^{h(m)} \tag{18}$$

$$= K(t). \tag{19}$$

Eq. (15) holds and GDEC outputs m correctly.

Considering C is poisoned i.e. $m' \neq m$, we claim Eq. (15) does not hold and GDEC outputs $\perp$. If this is not the case, we have

$$e = T^{h(m')}/T^{h(m)} \tag{20}$$

$$= g^{t(h(m')-h(m))} \tag{21}$$

Since $t, h(m) \in \mathbb{Z}_p^*$, $T = g^t$ is also a generator of $\mathbb{G}$, we know

$$h(m') - h(m) = 0 \mod p, \tag{22}$$

$$1 - p \leq h(m') - h(m) \leq p - 1. \tag{23}$$

Accordingly we have $h(m') = h(m)$ which contradicts with the collision-resistance of h. Therefore, GDEC would output $\perp$ in case C is poisoned. □

Theorem 2. *The key verification of MLEvd is **correct** in the sense that KV outputs 1 if and only if the key input is correctly computed based on $C = (c, T)$ and $F = g^{h(m)}$, i.e. $K = T^{h(m)}$.*

Proof. We first prove sufficiency. Assuming K is correct, due to the bilinearity we have

$$e(g, K) = e(g, T)^{h(m_i)} \tag{24}$$

$$= e(g^{h(m)}, T) \tag{25}$$

$$= e(F, T). \tag{26}$$

Thus KV outputs 1.

Next, we prove necessity. Given $T \in \mathbb{G}$ and T is not the identity element, it is easy to verify T is a generator of the group since the group size p is a prime.[3] Let $K = T^x$ where $x \in \mathbb{Z}_p^*$. If KV outputs 1, Eq. (16) holds, thus we have

$$e^\star = e(g, K)/e(F, T) \tag{27}$$

$$= e(g, T)^{x - h(m)}. \tag{28}$$

Since $x, h(m) \in \mathbb{Z}_p^*$, we know $x - h(m) = 0$, and

$$K = T^{h(m)}. \tag{29}$$

□

D. Security analysis

MLEvd achieves the semantic security when messages are unpredictable, i.e. have high min-entropy. Specifically, following the definition of PRV-CDA-security and the One-time Real-or-Random security defined in [8], we can prove

Theorem 3. *The MLEvd is **PRV-CDA-secure** in the random oracle model assuming the symmetric encryption scheme SDE satisfies one-time real-or-random security and CDH assumption holds on $\mathbb{G}$.*

The above theorem can be proved by using a hybrid argument and replacing the outputs of hash functions as well as the symmetric encryptions with random bits similarly as the proof of PRV$-CDA security in [8] does. The detailed proof is too lengthy to be included here. We leave it to the extended version.

VI. DTEp: A FULLY SECURE RE PROTOCOL THWARTING POISONING ATTACKS

In this section, we present our complete RE protocol DTEp. The DTEp shares similar procedures with DTEs, but uses MLEvd as its encryption scheme and has additional verification steps to deal with poisoning attacks.

A. Protocol Details

1) Chunking: DTEp's chunking is the same as DTEs'.

2) Key and fingerprint generation: To support efficient look-up operations, DTE-p uses a fixed fingerprint F_i:

$$F_i = g^{h(m_i)}. \tag{30}$$

For encryption/decryption key, David needs to be able to verify whether $K_i(t)$ and F_i match. But he cannot know how to compute $K_i(t)$ from F_i. Otherwise, David can compute the key by itself and decrypt all ciphertexts. Thus, we let

$$K_i(t) = g^{h(m_i)t}. \tag{31}$$

[3] In practice, we require the agent rejects T that is not a valid group member or is the identity element.

3) Deduplication: DTEp's deduplication is the same as DTEs'. For each chuck, Alice checks whether it can be deduplicated by looking up its corresponding fingerprint in a fingerprint table T_{La} stored in her local cache. Alice generates a deduplicated message $M' = \{m'_1, m'_2, \ldots, m'_n\}$ as follows.

$$m'_i = \begin{cases} m_i & \text{if } F_i \notin T_{La}; \\ h(m_i) & \text{otherwise.} \end{cases} \tag{32}$$

4) End-to-end secure transmission: DTEp's end-to-end secure transmission is the same as DTEs'.

5) Message recovery: DTEp's message recovery is similar to DTEs'. The only difference is Bob now needs to compute the fingerprint and secret key for each deduplicated chunk. Based on received $h(m_i)$, Bob first recovers the fingerprint F_i following Eq. (30). Then Bob sends F_i to David, and requests its corresponding ciphertext $C_i(t)$.

Note that to support dynamic keys, DTEp's ciphertext includes an additional nonce part as

$$C_i(t) = (c_i(t), T), \tag{33}$$

where

$$c_i = ENC(m_i, K_i(t)) \tag{34}$$

$$T = g^t \tag{35}$$

After receiving $C_i(t)$, Bob computes the secret key $K_i(t)$ as:

$$K_i(t) = (T)^{h(m)}, \tag{36}$$

and decrypts $c_i(t)$ with it

$$m_i = DEC(c_i(t), K_i(t)). \tag{37}$$

Finally, to assure the recovered message is correct, Bob can compute the fingerprint of the recovered message and compare it with the fingerprint that he received from Alice. If the two are the same, the message is correct. Otherwise, the recovery is false, and Bob requests Alice to send the correct message without performing deduplications.

6) Synchronizations: DTEp's synchronization is the same as DTEs'.

7) Poisoned ciphertext verification and correction: If Bob sees a poisoned ciphertext $C_i(t)$, he submits $< F_i, K_i(t) >$ to David.

David verifies if the following equation holds

$$e(F_i, T) = e(g, K_i(t)). \tag{38}$$

If it holds, David agrees that $K_i(t)$ is the correct key. David continues to verify the correctness of $\tilde{c_i}$ same as Bob does.

B. Correctness

Theorem 4. *DTEp's verification is **correct** in the sense that the deduplication agent always agrees when a honest user finds a poisoned ciphertext and reports it.*

Proof. Suppose a poisoned attack does happen and is reported by a honest user, the agent would receive the correct $K_i(t)$ corresponding to F_i and T:

$$K_i(t) = T^{h(m_i)} \tag{39}$$

$$= g^{h(m_i)t} \tag{40}$$

Due to the bilinearity, we have

$$e(g, K_i(t)) = e(g, g)^{h(m_i)t} \tag{41}$$

$$= e(g^{h(m_i)}, g^t) \tag{42}$$

$$= e(F_i, T). \tag{43}$$

Therefore, equation (38) holds and the agent agrees $K_i(t)$ submitted by the user is correct. After applying $K_i(t)$ for decryption on the poisoned ciphertext, agent would get a message m'_i that is different from m_i. Due to the collision-resistance of H, we know $H(m'_i) \neq F_i$. Thus, the agent would agree $C'_i(t)$ is poisoned. □

C. Security analysis

Theorem 5. *DTEp is **secure** against the agent in the sense that the agent cannot differentiate deduplicated messages of equal lengths assuming they are unpredictable, i.e. have high mini-entropy and the MLEvd is secure.*

Proof. We prove this in the random oracle model by examining what the deduplication agent sees in DETp.

For each deduplicated message m_i, if the ciphertext is correct, the agent would receive its fingerprint F_i and ciphertext $C_i(t)$, which are included in the adversary's view in MLEvd's security game. Since MLEvd is secure, the agent cannot differentiate deduplicated messages when they are unpredictable. The theorem holds.

Now consider a ciphertext is poisoned and a honest user reports it, the agent would receive its fingerprint F_i, its correct encryption key $K_i(t)$ and the nonce g^t. In addition, the agent may receive a new valid ciphertext $C_i(t')$ corresponding to a new nonce t'. However, knowing an extra, valid random $(K_i(t), g^t)$ pair does not give adversary any extra advantage (since the adversary can generate the pair by itself), therefore the theorem still holds. □

VII. Experimental Evaluations

In this section, we implement our protocol and perform experiments with real-world data to demonstrate the performance and efficiency of the MLEvd scheme and the DTEp protocol.

A. Environments and datasets

All experiments are performed on a computer with the Intel(R) Xeon(R) Gold 5122 CPU, 128 GB memory running the Ubuntu OS. We implement the MLEvd scheme and DTEp protocol using the Crypto++ Library [21], the PBC (Pairing-Based Cryptography) Library [22] and the The GNU Multiple Precision Arithmetic (GMP) Library [23] with C++ language. In MLEvd, we use ChaCha stream cipher [24] as the underlying deterministic encryption scheme, and SHA256 as the underlying cryptographic hash function. The pairing

algorithm we use is randomly generated by the PBC Library according to the security parameter. We use the Rabin CDC algorithm for the chunking scheme.

We use the Web-site hosts traces and snapshots collected by the File systems and Storage Lab and its collaborators [25], [26]. We transmit files derived from the homes snapshots in 2015 between the senders and receivers through the servers in our experiments and collect the experimental results during the transmissions. The results are shown in the following.

For simplicity, we implement our protocols in the application layer. In practice, one can implement our protocols in lower layers such as transport layer or network layer which should be able to further increase the transmission efficiency.

B. Results on bandwidth saving

We evaluate the bandwidth savings by comparing the actual size of messages transmitted between the sender and the receiver to the original size of messages, i.e.

$$bandwidth\ saving = 1 - \frac{actual\ size\ of\ messages}{original\ size\ of\ messages}.$$

1) Using different chunking sizes: Figure 2 shows DTEp's bandwidth saving under different chunking sizes. We record the DTEp's performance while the sender and the receiver constantly transmitting 3 random files in 7 consecutive periods. The chunking sizes are the average bit length of the chunks. The experiments are conducted with 256 bits security parameter and 32 CPU cores. The size of all caches of senders, receiver and servers are set as 128 MB, and we totally transmit messages of size about 2 GB. The cache replacement policy is set to FIFO. The experimental results show that with larger average chunking size, the bandwidth savings are evidently improved. This is mainly because when deduplication happens, the chunk contents will replaced by its hash in the message. So larger chunks can bring more savings when deduplications happen. When using 8KB/16KB chunks, DTEp can save up to 60-70% bandwidth.

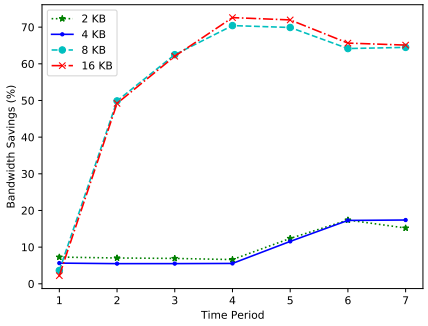

Fig. 2. Bandwidth Savings with Different Chunking Sizes

2) Using different cache sizes and replacement policies: We test three commonly-used kinds of cache replacement policies namely the "first in first out (FIFO)" policy, the "least recently used (LRU)" policy and the "least frequently used (LFU)" policy, with cache sizes ranging from 4 MB to 128 MB. The bandwidth savings are shown in Figure 3. We can see that when the cache size is small, the bandwidth savings are not large, increasing the size could result greater bandwidth savings. On the other hand, when the cache capacity is sufficiently large, further increasing cache is not very helpful. In addition, we see different cache replacement policies do not obviously affect the bandwidth savings on our datasets.

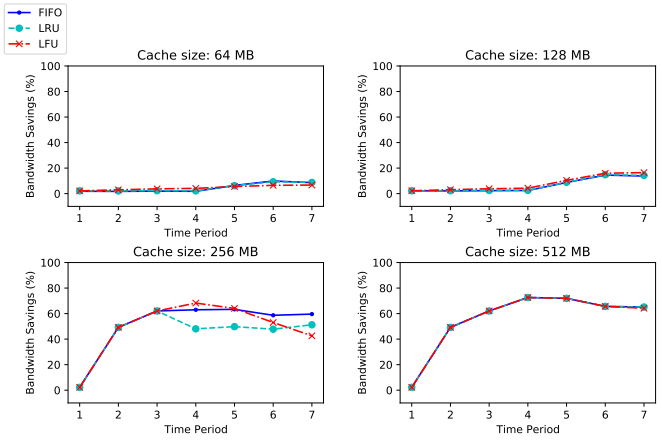

Fig. 3. Bandwidth Savings with Different cache Settings

C. Results on efficiency

Then we conduct the following experiments aiming to show that the overheads brought by the DTEp protocol is lightweight. In these experiments, we evaluate the processing throughput, i.e. how much data can be processed per second by the sender and receiver, under different parameters. We also record the throughput needed for the sender of messages to update the cache of the server. The experimental data presented here is collected synchronously with the bandwidth savings shown above. So the experimental results also shows the correlation of the bandwidth savings and the processing throughput.

1) Using different security parameters: In Figure 4, we show the processing throughput under different security parameters for the MLEvd scheme. The experimental results meet the expectation that with the throughput will decrease with greater security parameter, because more computation resource will be consumed by the cryptographic operations. Our protocol can achieve roughly 200 Mbps throughput at both the sender's and the receiver's sides which is practical for WAN transmissions.

2) Using different cache sizes and replacement policies: In Figure 5 and Figure 6, we show how the cache replacement policy and the cache size affect the processing throughput. The cache replacement policy do not have much effect on the processing throughput. But larger cache size will improve the cache updating throughput but slower the receiving throughput. This is because that larger cache size causes more redundancy elimination, so the cache no longer needs too

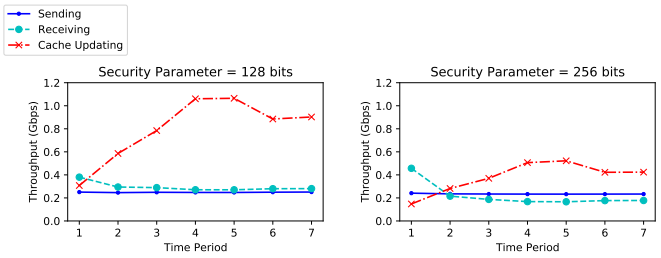

Fig. 4. Throughput with Different Security Parameters

much update. But at the same time, the receivers need to pay more effort to download the cache and decrypt the ciphertext in the cache.

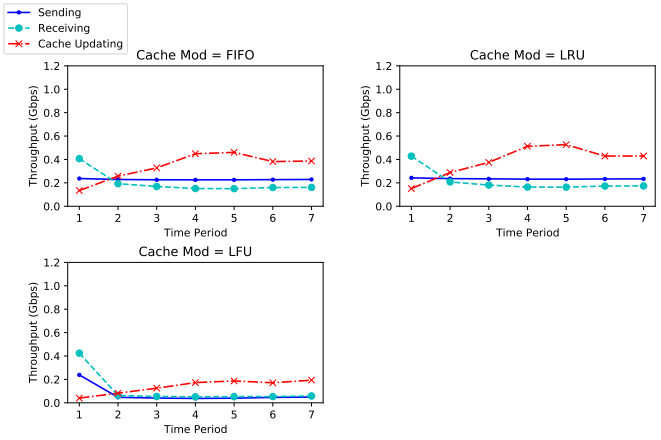

Fig. 5. Throughput with Different Cache Policies

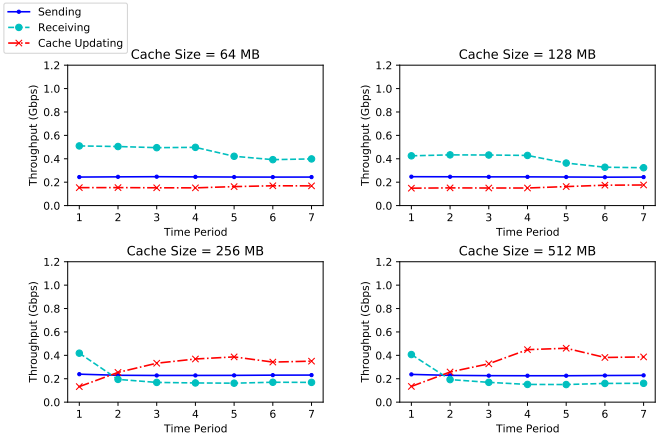

Fig. 6. Throughput with Different Cache Sizes

3) Using different chunking sizes and amounts of threads: We also show the effect of different average chunking sizes in Figure 7. Different chunking sizes mainly affect the updating throughput. The reason for this is that smaller chunking size means more chunks and this leads to more cache updating. We also present the throughput with different amounts of threads in Figure 8.

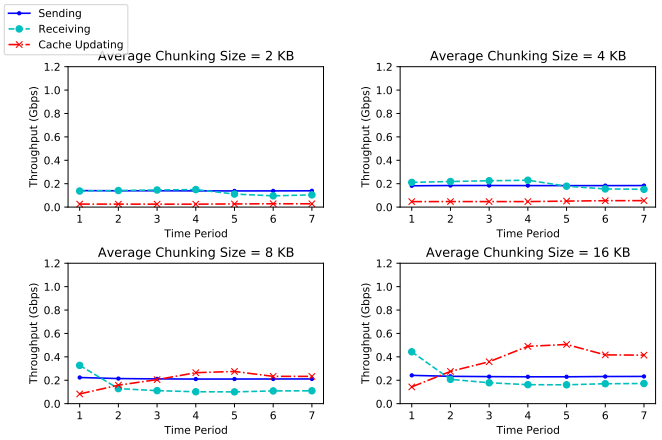

Fig. 7. Throughput with Different Chunking Sizes

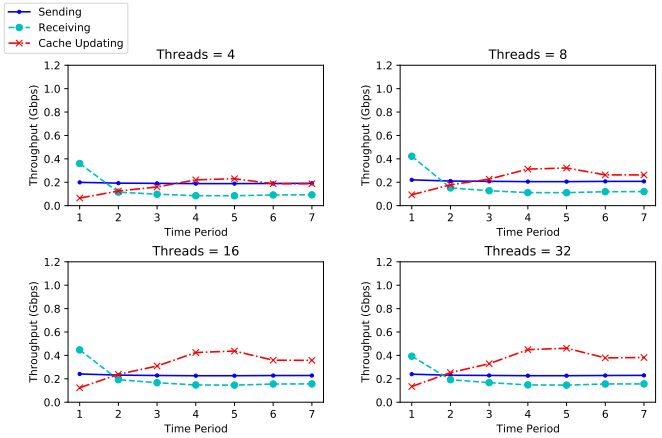

Fig. 8. Throughput with Different Amounts of Threads

VIII. Conclusion

In this paper, we study the problem of designing secure cross-user RE systems for WAN optimization. Different from existing RE works which break the end-to-end security or focus on single-user scenario, we propose a novel RE protocol that preserves the end-to-end security and can thwart poisoning attacks in the cross-user setting for the first time. Based on our strategies of "deduplicate-then-encrypt" and "key revealing for ciphertext verification" and our novel MLEvd scheme, we construct a highly efficient, secure RE protocol. Experimental results show our protocol can achieve up to 60%-70% bandwidth savings and around 200 Mbps throughput using a preliminary application-layer implementation. Our future work include implementation optimizations of the MLEvd and DTEp, and real-world WAN tests.

REFERENCES

[1] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000, pp. 87–95.

[2] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: findings and implications," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, 2009, pp. 37–48.

[3] S. Ihm and V. S. Pai, "Towards understanding modern web traffic," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 295–312.

[4] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage (ToS)*, vol. 7, no. 4, pp. 1–20, 2012.

[5] "Cisco wide area application services ssl application optimizer deployment guide," https://www.cisco.com/c/en/us/products/collateral/routers/wide-area-application-services-waas-software/deployment_guide_c07-541981.html, 2019, accessed: 2021-3-28.

[6] J. Fan, C. Guan, K. Ren, and C. Qiao, "Middlebox-based packet-level redundancy elimination over encrypted network traffic," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 4, pp. 1742–1753, 2018.

[7] B. Agarwal, A. Akella, A. Anand, A. Balachandran, and G. Varghese, "Endre: An end-system redundancy elimination service for enterprises," in *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010, April 28-30, 2010, San Jose, CA, USA*, 2010, pp. 1–12.

[8] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2013, pp. 296–312.

[9] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proceedings 22nd international conference on distributed computing systems*. IEEE, 2002, pp. 617–624.

[10] J. Xu, E.-C. Chang, and J. Zhou, "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 195–206.

[11] Q. Huang, Z. Zhang, and Y. Yang, "Privacy-preserving media sharing with scalable access control and secure deduplication in mobile cloud computing," *IEEE Transactions on Mobile Computing*, 2020.

[12] X. Yang, R. Lu, J. Shao, X. Tang, and A. Ghorbani, "Achieving efficient secure deduplication with user-defined access control in cloud," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[13] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. Deng, "Secure cloud data deduplication with efficient re-encryption," *IEEE Transactions on Services Computing*, 2019.

[14] Y. Zhang, Y. Mao, M. Xu, F. Xu, and S. Zhong, "Towards thwarting template side-channel attacks in secure cloud deduplications," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[15] M. W. Storer, K. Greenan, D. D. Long, and E. L. Miller, "Secure data deduplication," in *Proceedings of the 4th ACM international workshop on Storage security and survivability*, 2008, pp. 1–10.

[16] A. Agarwala, P. Singh, and P. K. Atrey, "Dice: A dual integrity convergent encryption protocol for client side secure data deduplication," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2176–2181.

[17] M. O. Rabin, "Fingerprinting by random polynomials," *Technical report*, 1981.

[18] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 174–187.

[19] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang, "Fastcdc: a fast and efficient content-defined chunking approach for data deduplication," in *2016 {USENIX} Annual Technical Conference ({USENIX}{ATC} 16)*, 2016, pp. 101–114.

[20] W. Xia, X. Zou, H. Jiang, Y. Zhou, C. Liu, D. Feng, Y. Hua, Y. Hu, and Y. Zhang, "The design of fast content-defined chunking for data deduplication based storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2017–2031, 2020.

[21] W. Dai, "Crypto++ library," https://www.cryptopp.com/.

[22] B. Lynn, "Pbc (pairing-based cryptography) library," https://www.cryptopp.com/.

[23] T. Granlund, "Gnu multiple precision arithmetic library," https://gmplib.org/.

[24] D. J. Bernstein *et al.*, "Chacha, a variant of salsa20," in *Workshop record of SASC*, vol. 8, 2008, pp. 3–5.

[25] V. Tarasov, A. Mudrankit, W. Buik, P. Shilane, G. Kuenning, and E. Zadok, "Generating realistic datasets for deduplication analysis," in *2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12)*, 2012, pp. 261–272.

[26] F. systems and S. Lab, "Traces and snapshots public archive," https://tracer.filesystems.org/.