Distributed Key Generation for Secure Encrypted Deduplication

Yitao Duan

NetEase Youdao
Beijing, China
duan@rd.netease.com

Abstract

Large-scale storage systems often attempt to achieve two seemingly conflicting goals: (1) the systems need to reduce the copies of redundant data to save space, a process called deduplication; and (2) users demand encryption of their data to ensure privacy. Conventional encryption makes deduplication on ciphertexts ineffective, as it destroys data redundancy. A line of work, originated from Convergent Encryption [28], and evolved into Message Locked Encryption [12], strives to solve this problem. The latest work, DupLESS [11], proposes a server-aided architecture that provides the strongest privacy. The DupLESS architecture relies on a key server to help the clients generate encryption keys that result in convergent ciphertexts. In this paper, we first provide a rigorous proof of security, in the random oracle model, for the DupLESS architecture which is lacking in the original paper. Our proof shows that using additional secret, other than the data itself, for generating encryption keys achieves the best possible security under current deduplication paradigm. We then introduce a distributed protocol that eliminates the need for a key server and allows less managed systems such as P2P systems to enjoy the high security level. Implementation and evaluation show that the scheme is both robust and practical.

1 Introduction

Deduplication is a very important technique that many storage systems use to reduce cost. It exploits the redundancy in data and avoids having to store identical data multiple times. It is reported that effective deduplication can achieve up to 50% or even 90% saving on disk space and bandwidth [33, 18, 28, 22]. Deduplication has seen wide-spread use and been built into many production cloud storage systems such as Dropbox [3], EMC [38], etc., as well many Peer-to-Peer (P2P) systems such as [56, 57, 5, 6].

Deduplication, however, comes in direct conflict with another crucial goal of storage systems: data privacy which is provided via encryption. Deduplication exploits the redundancy that exists in naturally generated data (e.g., images, documents) and by nature leaks (to the server) the equality information among data chunks. Conventional encryption regards this as a serious leakage and destroys the redundancy, making deduplication ineffective. Convergent Encryption (CE) [28] was proposed as a workaround. With CE, a piece of data is encrypted with a key derived deterministically from its content. This guarantees a convergence property: the same chunk always encrypts to the same ciphertext. Although lacking a rigorous theoretical treatment on its security, CE, or its variants, has been deployed in many research [13, 7], open-source and commercial systems such as Freenet [5], GNUnet [6], flud [4], Ciphertite [2], Bitcasa [1], and [55].

The protection CE provides, however, is too weak. The content-based key generation process allows anyone who has access to the data to derive the key and perform encryption. This makes it vulnerable to off-line brute-force dictionary attack. Subsequent works result in a few variants but do not eradicate the vulnerability. Very recently, a USENIX Security '13 paper introduced the DupLESS [11] approach that includes an additional secret in the encryption key generation process. This disables dictionary attacks. To ensure the convergence property, this secret is identical for all users and provided via a key server (KS) in DupLESS. As we will show in this paper, DupLESS provides the best possible security for encrypted deduplication. However, its reliance on a dedicated key server makes it difficult to deploy in a less managed setting such as P2P systems.

Our work originated independently of, and prior to the publication of, [11]. [1] We took a similar approach as we also include a global secret in the key generation. However, unlike [11], our scheme is distributed and does not need the key server. We also provide a rigorous security analysis which is lacking in [11]. Specifically, our contributions include

- A new notion of security, denoted D-IND$ for deterministic indistinguishability from random strings, with proof that it is strictly stronger than all existing relevant notions.
- A rigorous proof that DupLESS architecture is D-IND$-CPA secure in the random oracle model.
- A distributed architecture for encrypted deduplication, achieving D-IND$-CPA security. The scheme is robust against the collusion between the server and up to a threshold number of users.

Our scheme features simplified and scalable key management: each party involved only needs to maintain a key of constant length, independent of the number of users or the number of data blocks. It facilitates efficient sharing: if desired, users within a group can freely share their data without having to acquire additional keys. This is not possible with DupLESS. We also make connections to research in statistical database security and point out possible research directions that could measure the actual privacy implication of even the best encrypted deduplication. We believe this is a crucial step towards a holistic understanding of privacy issues caused by deduplication.

We have implemented our protocol and evaluated its performance. Our tests show that the scheme is practical for many real-world applications. For example,

[1] We became aware of the work of [11] after the USENIX Security '13 conference.

at a a quite strong robustness level, our protocol can retain 55% to 90% of the original system upload throughput, depending on the configurations, while achieving 35% space saving.

2 Related Work

Convergent encryption [28] was the first popular solution to encrypted deduplication. The scheme is very simple: data is encrypted using a symmetric encryption scheme with a key derived deterministically (e.g., via a hash function) from its content. It has been deployed in many systems [13, 7, 5, 6, 4, 2, 1, 55].

Theoretical analysis of encrypted deduplication was then conducted in the context of deterministic encryption. [8] defines a semantic-security style notion of security, denoted PRIV security, for a deterministic public-key encryption scheme (D-PKE) and gives constructions in the random oracle model. Two follow-up works [17, 10] provided definitional equivalences and constructions without random oracles. This line of work was recently extended to the symmetric encryption setting by the Message-Locked Encryption (MLE) framework [12] where the key used for encryption and decryption is itself derived from the message. CE is analyzed under this framework in [12]. [12] also introduces a few new security notions that are similar to the PRIV security in [8].

The Problem with Public Encryption. It is interesting to notice that, although differing in their key distribution configurations (MLEs are symmetric [12] while D-PKEs are asymmetric or public-key [8, 17, 10]), the encryption processes in both MLE and D-PKE are *public*, meaning that anyone having access to the data can perform the encryption and produce valid ciphertext. The reason is probably due to the need for each player to produce converging ciphertexts independently. This design has very serious security consequences:

1. The schemes can only protect unpredictable messages. Brute-force attack can recover messages with low entropy. In reality, however, data is often predictable [11].
2. Even if the message source has high entropy, the adversary can easily obtain one bit of information: whether a ciphertext is the encryption of a message from a small set. This can be a serious leakage. For example, one could imagine the service provider's dilemma when it is approached by RIAA or MPAA with a list of files and demanding that they refuse to host any content matching those files, or identify the users with matching data.

A very recent work, DupLESS [11], attempts to solve the problem. It introduces another secret key, denoted SK, held by a key server. Data is encrypted with a key that is derived from both the data and SK. Encryption is thus not public anymore. We show later that this achieves the strongest possible security for deduplication encryption.

3 Preliminaries

Deduplication can be either at file level or block level. Our scheme is oblivious to this differentiation and in this paper we refer to the units of deduplication (file or block) as *chunks*. Our scheme can be applied to the scenario either with or without a storage service provider (denoted the server). In the latter case the data can be scattered in a P2P fashion among the users. Our system can tolerate up to t corrupted users where t is a system parameter. Corrupted users are allowed to collude with each other and with the server if there is one.

We use $\mathbb{Z}$ to denote the set of integers. For a positive integer N, let $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ and $\mathbb{Z}_N^* = \{i \in \mathbb{Z} : 1 \leq i \leq N-1 \quad \text{and} \quad \gcd(i, N) = 1\}$. For a vector v we write $v[i]$ to denote its ith element. The notation $s \overset{\$}{\leftarrow} S$ means that an element is uniformly randomly selected from the set S and assigned to the variable s. We use boldface, e.g., **SE**, to denote an encryption scheme and, without causing confusion, we also use the same notation for its encryption algorithm. That is, $\mathbf{SE}_K(M)$ means the encryption of message M with key K by the algorithm **SE**. Since we focus on the privacy of encryption in this paper, we ignore decryption algorithms whenever convenient. They should be assumed to have the standard properties such as efficient recovery of plaintext given the ciphertext and the key.

4 A New Notion of Security

Research in cryptography has established the importance of a proper notion of security. Over the years many ad hoc and syntactic notions that target at certain properties that "seem" right have been proven inadequate. In 1982 Goldwasser and Micali proposed the notion of semantic security [41]. A cryptosystem is semantically secure if any probabilistic, polynomial-time adversary that is given the ciphertext of a certain message, and the message's length, cannot determine any partial information on the message with probability non-negligibly higher than all other probabilistic, polynomial-time algorithm that only have access to the message length (but not the ciphertext) [41]. Later Goldwasser and Micali proved that semantic security is equivalent to ciphertext indistinguishability (IND) [42] which is easier to work with. Semantic security or IND is a strong property and can be combined with different attack scenarios, such as chosen plaintext attacks (CPA), chosen ciphertext attack (CCA1), and adaptive chosen ciphertext attack (CCA2), to form various security levels.

Clearly any MLE or D-PKE cannot be semantically secure, as they leak message equality. We introduce a new notion of security, denoted D-IND$ for deterministic indistinguishability from random strings, which we show to be the strongest among all relevant notions [8, 51, 10, 12]. We then proceed to prove that this is the best achievable security under current deduplication paradigm: the scheme does not allow a computationally bounded adversary to extract more information about user data besides what is necessary for deduplication.

4.1 D-IND$

The new security notion is an extension to a similar notion for randomized symmetric encryption which we state in the following. The notion is called indistinguishability from random strings (denoted IND$ in this paper). It is introduced in [52] and captures an adversarys inability to distinguish a ciphertext from a random string of the same length. This is a particularly strong notion and can be shown to imply more standard definitions such as find-then-guess, left-or-right, and semantic security etc [52, 9]. We focus on its CPA version. Others such as CCA and adaptive versions follow standard extensions.

IND$. For a one-time symmetric encryption scheme $\mathbf{SE}$ with key space $\{0,1\}^k$, an adversary A's IND$-CPA advantage is defined with respect to the following game: a random bit b is drawn. Then adversary A is run. A can make multiple queries to an encryption oracle **Enc**. On each query M, if $b = 1$, **Enc** will first choose a random key $K \xleftarrow{\$} \{0,1\}^k$ and return $C \leftarrow \mathbf{SE}_K(M)$. If $b = 0$, **Enc** simply returns $C \xleftarrow{\$} \{0,1\}^{\mathrm{cl}_{\mathbf{SE}}(k,|M|)}$ where $\mathrm{cl}_{\mathbf{SE}}(k, |M|)$ is the ciphertext length if M is encrypted by $\mathbf{SE}$ with security parameter k. A should output a bit b' at the end of the game. The advantage of A is defined as

$$\mathbf{Adv}_{\mathbf{SE}}^{\mathrm{IND\$-CPA}}(A) = \Pr[b' = b] - 1/2$$

D-IND$. The definition of deterministic indistinguishability from random strings under CPA, denoted D-IND$-CPA, is identical to that of IND$-CPA except that the adversary is restricted to querying the encryption oracle with only *distinct* messages. This restriction is inherit with deterministic encryption as deterministic encryption inevitably leaks message equality. In [8, 10] this is formalized as requiring the two message sequences submitted to the encryption oracle to have the same quality pattern. Distinct message sequence is easier to work with and it has been shown to be equivalent to the identical equality pattern restriction [51]. Intuitively, with a deterministic encryption, an adversary gains nothing by repeatedly querying the oracle with the same message.

It is worth pointing out that, similar to its counterpart for standard, randomized symmetric encryption, IND$, the definition of D-IND$ does *not* place any restriction on the message entropy. This is in contrast to the notions for D-PKE [8, 10] and MLE [12] which only protects unpredictable messages.

4.2 Relating to Other Notions

Other notions of security for D-PKE or MLE, which are relevant to encrypted deduplication, have been introduced in [8, 10] and [12]. Among them, PRV$-CDA is the strongest and implies all others, including the adaptive versions [12]. Now we show that D-IND$-CPA is strictly stronger than PRV$-CDA. PRV$-CDA is introduced in [12] in the context of MLE whose encryption is public while in the D-IND$-CPA model it is often not the case so we need to adapt PRV$-CDA in order for the two to be comparable. The adaption is very simple: substituting the encryption in MLE with an encryption oracle suffices to allow the adversary to work in both models without affecting any information it obtains.

Following the standard approach for analyzing relations among security notions [9], we show that any scheme that is D-IND\$-CPA secure is also PRV\$-CDA secure but the converse is not true. To prove this, we show that, any adversary's PRV\$-CDA advantage is bounded in someway by some adversary's D-IND\$-CPA advantage. The existence of a counterexample, i.e., there exists a scheme that is PRV\$-CDA secure but is not D-IND\$-CPA secure, is sufficient to prove the other direction.

PRV\$-CDA Security. The essence of PRV\$-CDA security is summarized as follows. For detailed discussion please see [12]. Ignoring the tagging algorithm, an MLE scheme $\mathit{\Pi} = (\mathbf{P}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ consists of the following polynomial time algorithms: public parameter generation $\mathbf{P}$, key generation $\mathbf{K}$, encryption $\mathbf{E}$ and decryption $\mathbf{D}$. A source $\mathcal{M}$ is a polynomial time algorithm that on input 1^k generates $(\mathbf{M}, z)$ where $\mathbf{M} \in (\{0,1\}^*)^q$ is a vector of q messages with each $\mathbf{M}[i] \in \{0,1\}^*$ and $z \in \{0,1\}^*$. There is the usual restriction for MLE adversaries that the entries of $\mathbf{M}$ are all distinct. Unless otherwise stated, all operations on vectors are element-wise. The PRV\$-CDA advantage of adversary A is defined as

$$\mathbf{Adv}_{\mathit{\Pi}}^{\mathrm{PRV\$-CDA}}(A) = \\ \Pr\left[\begin{array}{l} P \xleftarrow{\$} \mathbf{P}(1^k); b \xleftarrow{\$} \{0,1\}; \\ (\mathbf{M}, z) \xleftarrow{\$} \mathcal{M}; \mathbf{C}_1 \xleftarrow{\$} \mathbf{E}_P(\mathbf{K}_P(\mathbf{M}), \mathbf{M}); \\ \mathbf{C}_0[i] \xleftarrow{\$} \{0,1\}^{|\mathbf{C}_1[i]|}, \forall i \in \{1, \ldots, q\}; \\ b' \leftarrow A(P, \mathbf{C}_b, z) : b = b' \end{array}\right] - 1/2$$

Theorem 1 (D-IND\$-CPA ⇒ PRV\$-CDA). *Let $\mathit{\Pi} = (\mathbf{P}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ be an MLE scheme. For any PRV\$-CDA adversary A against $\mathit{\Pi}$, there exists a D-IND\$-CPA adversary, B, also against $\mathit{\Pi}$, such that*

$$\mathbf{Adv}_{\mathit{\Pi}}^{\mathrm{PRV\$-CDA}}(A) \leq \mathbf{Adv}_{\mathit{\Pi}}^{\mathrm{D-IND\$-CPA}}(B)$$

Proof. B is constructed as follows. It runs $P \xleftarrow{\$} \mathbf{P}(1^k)$ and $(\mathbf{M}, z) \xleftarrow{\$} \mathcal{M}(1^k)$ to get P and $(\mathbf{M}, z)$. It then feeds $\mathbf{M}$ to its oracle. Let $\mathbf{C}$ be what is returned by the oracle. B runs A with $(\mathbf{C}, z)$. Note that since A is restricted to asking only distinct queries, B does not need to do anything special with the queries or returned ciphertexts. When A terminates and returns a bit b', B outputs the same bit.

During the process, $\mathbf{C}$ is either the encryption of $\mathbf{M}$ or some random bit strings with the same lengths. A get identical information as if it is attacking $\mathit{\Pi}$ in a PRV\$-CDA game. The advantage (success probability) of B is at least that of A.

Theorem 2 (PRV\$-CDA ⇏ D-IND\$-CPA). *There exists a scheme that is PRV\$-CDA secure but not D-IND\$-CPA secure.*

This is trivial to show as CE and all MLEs that are proven to be PRV$-CDA secure in [12] are not D-IND$-CPA secure: their public encryption nature allows the adversary to distinguish encryptions of **M** from random strings by encrypting the messages and comparing **C** with the resulting ciphertexts.

5 Security of EwS

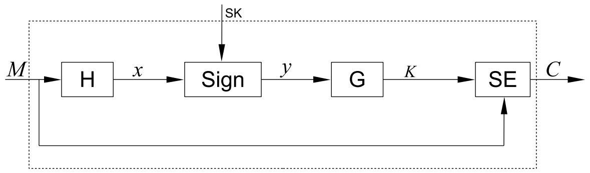

Fig. 1. Encryption with Signature

We now analyze the security of DupLESS architecture. Conceptually, the encryption process of DupLESS can be depicted by the schematic in figure 1. H and G are two hash function, **Sign** is a signing algorithm generating signatures on input messages using private key SK. In DupLESS, SK is held by the key server. **SE** is an one-time IND$-CPA symmetric encryption scheme. The scheme generates the randomness for **SE** with the hash of the signature on the message. [2] We denote the scheme Encrypt-with-Signature (EwS).

The security of the EwS is based on the security of its two subcomponents: IND$-CPA security of **SE** and existential non-forgeability of **Sign**. We cast the latter in an adaptive chosen message attack (CMA) setting and define the advantage of an adversary to be the maximum probability that the adversary successfully forges a signature. To make it concrete, we provide the proof for an instantiation of the scheme using deterministic RSA-based signature. This is also the choice of DupLESS [11]. The proof framework is actually more general and can work through as long as the signature scheme has a simulator in the random oracle model that generates valid signatures on given messages by manipulating hash oracle(s).

Let p and q are two random primes of equal length (e.g., 512 bits) with $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' themselves are primes. [3] The RSA

[2] DupLESS uses blind signature to prevent the key server from learning any information about the data (e.g., its hash $H(M)$). Given the information-theoretic privacy nature of blind signature [20], this does not affect the security of the encryption scheme thus omitted from the analysis.

[3] This is actually stronger than the requirements for standard RSA signature. This property is necessary when we introduce the distributed version in section 6.2.

modulus is $N = pq$. Let the RSA public exponent be a prime $e > N$. [4] The public key is $PK = (N, e)$. The private key is (d, N) where $ed \equiv 1 \mod \phi(N)$. H is assumed to be $H : \{0,1\}^* \to \mathbb{Z}_N^*$. The standard RSA signature on message M is $y \in \mathbb{Z}_N^*$ such that $H(M) = y^e$.

Same as [12], the underlying symmetric encryption is assumed to be IND\$-CPA secure as defined in section 4.1. AES with standard CTR mode and fixed IV, and the OCB Authenticated Encryption scheme introduced in [52] are all such schemes. In addition, and this is also in line with [12], the scheme is a one-time symmetric encryption, meaning that it generates a fresh key for each message.

Theorem 3. *Let* **SE** *be a one-time symmetric encryption scheme whose key space is* $\{0,1\}^l$ *and* **Sign** *a signature scheme. Let* $\mathbf{EwS}(1^k)$ *be an EwS encryption scheme constructed as in figure 1 using* **SE** *and* **Sign**. *For any adversary A attacking the D-IND\$-CPA security of* **EwS** *that runs in time at most t and asks at most q_e encryption oracle calls and q_H (resp. q_G) queries to H (resp. G), with these queries totaling at most μ bits, there exists an adversary B attacking the standard IND\$-CPA security of* **SE** *such that*

$$\mathbf{Adv}_{\mathbf{EwS}}^{\mathrm{D-IND\$-CPA}}(A) \leq \mathbf{Adv}_{\mathbf{SE}}^{\mathrm{IND\$-CPA}}(B) + q_G \mathbf{Adv}_{\mathbf{Sign}}^{\mathrm{Forge-CMA}}(A) \qquad (1)$$

where $\mathbf{Adv}_{\mathbf{Sign}}^{\mathrm{Forge-CMA}}(A)$ *is the advantage A has in attacking the existential non-forgeability of* **Sign**. *Furthermore, B makes q_e encryption oracle queries and its running time is $t + O(\mu)$.*

Proof. B is constructed as the algorithm depicted in figure 2. It runs A against a simulated EwS constructed using **SE** that B tries to attacks.

Let Λ_H and Λ_G be the sets of queries A submits to H and G, respectively, and Λ_{Sign} the set of messages A submits to the signing oracle. Consider the following event F: there exists a pair (x, M) such that $x \in \Lambda_G$ and $M \in \Lambda_H$ but $M \notin \Lambda_{Sign}$ and $H[M] = x^e \mod N$. In other words, this is the event when A manages to succeed in forging a signature. We argue that F is the only case when A's view in the simulated EwS game differs from that of an actual EwS attacking game. To see this, suppose F does not happen, then for any query M, what A obtains from **Enc** is either random strings (if $b = 0$) or, if $b = 1$, encryption of M by **SE** under some key K which is randomly generated via $K \xleftarrow{\$} \{0,1\}^l$. In the latter case, the "correct" response A expects, in attacking EwS, is $\mathbf{SE}_{K'}(M)$ where $K' \leftarrow G[\mathbf{Sign}_{SK}(M)]$. Instead, B provides A with $\mathbf{SE}_K(M)$. This is equivalent to setting $K \leftarrow G[\mathbf{Sign}_{SK}(M)]$ in the EwS scheme. B does not know K but it does not matter: If G has never been queried with a valid signature on M, B is free to choose any random number in the range of G. Since K is randomly generated in its game against **SE** and independent of other keys, it is a perfectly valid value for $G[\mathbf{Sign}_{SK}(M)]$.

[4] The purpose of such a property is, together with e being a prime, to ensure $gcd(\phi(N), e) = 1$ even if N is not properly generated. This in turn guarantees that the blinding technique works.

```
**Adversary** $B^{\mathbf{Enc}(1^k,b)}(1^k)$
Run $A$ on input $1^k$, replying to its
oracle queries as follows:
**On Encryption Query** $\mathbf{EwS}(M)$:
  Forward $M$ to its encryption oracle **Enc**
  and obtains $C \leftarrow \mathbf{Enc}(M)$
  Return $C$
**On Query** $H(M)$:
  If $H[M]$ is undefined then
    $\mathtt{sigMap}[M] \overset{\$}{\leftarrow} Z_N^*$
    $H[M] \leftarrow \mathtt{sigMap}[M]^e \mod N$
  Endif
  Return $H[M]$
**On Query** $G(x)$:
  If $G[x]$ is undefined then
    $G[x] \overset{\$}{\leftarrow} \{0,1\}^l$
  Endif
  Return $G[x]$
**On Query** $Sign(M)$:
  If $\mathtt{sigMap}[M]$ is undefined then
    Query $H$ with $M$
  Endif
  Return $\mathtt{sigMap}[M]$
When $A$ returns a bit $b'$, return $b'$.
```

Fig. 2. Adversary B for theorem 3

And F is simply the success in attacking the existential non-forgeability of a signature scheme in an adaptive chosen message scenario whose probability is bounded by $\mathbf{Adv}_{\mathbf{Sign}}^{\mathrm{Forge-CMA}}(A)$. By simple union bound, we have equation 1. [5]

6 EwS without the Key Server

In this section introduce a distributed protocol that removes the need for a centralized key server. There are several advantages in such a change. First, even in a corporate setting, a dedicated key server represents additional resource requirement and can be a bottleneck for both security and performance. Security-wise, compromising a single host (the KS) reduces the security to the level of CE which has been shown to be inadequate [11]. As for performance, a single server may have inadequate QPS to serve a large number of clients. Duplicating the key server improves performance, at the price of higher hardware and operation cost, but does not help with the security issue, as each server still holds the same secret key SK. Furthermore, the DupLESS architecture is incompatible with the P2P

[5] Note that the advantage the adversary gains by querying H or the signing oracle multiple times is already factored into $\mathbf{Adv}_{\mathbf{Sign}}^{\mathrm{Forge}}(A)$.

paradigm as it is very difficult to deploy, maintain and secure a dedicated server in a P2P setting where users may be distributed over a wide range of geographic regions. Yet vulnerabilities exist in such systems that calls for tightened security. Many P2P systems, such as Freenet [5], flud [4], and GNUnet [6], are designed specifically for security, anonymity and/or censorship-resistance. They use CE for their protection. However, due to the weakness of CE, it is relatively easy to break their promises. For example, an attacker can mount many of the serious attacks outlined in [54] by compromising a *single* node.

We bridge the gap and present a decentralized version of encrypted deduplication that attains the same security as DupLESS. Our scheme is fully compatible with the P2P paradigm: in a P2P setting, a user may interact with different entities each time it performs certain operation (e.g., uploading a file). With our scheme, as long as the user obtains the cooperation of *any* qualified subset of players, it can perform the desired operation.

6.1 Threshold Signature

Our solution is built upon threshold signature scheme which we introduce in this section. Let n be the number of players and $t < n$ a parameter, a $(t+1, n)$-threshold signature scheme allows any subset of $t+1$ players to generate a signature, but prohibits any t or less players from producing a valid signature. Standard notions of security include non-forgeability and robustness.

Non-forgeability. Non-forgeability for a (non-interactive) threshold signature scheme is extended from the notion for standard signature [44] and is defined with respect to the following game. The adversary chooses to corrupt any fixed set of t players. Then the key generation algorithm is run. The public key, verification keys and private keys of the corrupted players are given to the adversary. The other private keys are given to the uncorrupted players. Next, the adversary submits signing requests to the uncorrupted players for messages of its choice. Upon such a request, a player outputs a *signature share* for the given message. An adversary forges a signature if at the end of the game it outputs a valid signature on message M that it has never submitted to any uncorrupted players as signing request. The scheme is non-forgeable if no polynomial time adversary could forge a signature.

Robustness. A threshold signature scheme should also guarantee that corrupted players should not be able to prevent uncorrupted players from generating signatures. Standard technique is to have each signer generate a “proof of correctness” that ensures that its signature share can be combined with other valid shares to produce the correct signature. The system typically generates additional verification keys for such purpose.

There are threshold signature schemes with provable non-forgeability and robustness. They can be based on either Diffie-Hellman problem [46] or RSA [53, 25, 39].

6.2 Distributed Oblivious Key Generation

In our decentralized EwS scheme, we replace the signature scheme with a threshold one. We call the scheme Distributed Oblivious Key Generation (DOPG). This is a nature fit both from a fault tolerant perspective and by the P2P principle: a client can accomplish its task as long as it can obtain the cooperation of *any* subset of qualified nodes. For our purpose we need an efficient, deterministic and non-interactive threshold signature. Shoup's RSA-based scheme [53] is such a solution. The deterministic nature of the scheme ensures the convergence property. It is also non-interactive, and is essentially as efficient as possible: the workload for generating a signature share is roughly equivalent to computing a single RSA signature.

We introduce a few modifications to the scheme. The first is an optimization trick that significantly boosts its performance. We then introduce a "blind" version that prevents the signers from obtaining information about the data. The scheme is summarized as follows.

Setup. Let n be the total number of signers. The RSA system parameters p, q, p', q', e, N and RSA public/private key pair are the same as specified in section 5 where we proved the security of centralized EwS. Let $m = p'q'$. The private key d is shared using a random degree t polynomial $f(X)$, over the ring $\mathbb{Z}_m$, whose free term is set to d. Player i's secret key is $s_i = f(i) \mod m$. Let Q_N be the subgroup of squares in $\mathbb{Z}_N^*$. The dealer chooses a random $v \in Q_N$ and computes $v_i = v^{s_i}$ for $i = 1, \ldots, n$. The verification key for player i is (v, v_i).

Signature Shares and Proof of Correctness. Given $x = H(M)$ for message M, let L_N be the bit length of N and L_1 a secondary security parameter (e.g., $L_1 = 128$), the signature share of player i is

$$x_i = x^{2\Delta s_i} \in Q_N, \quad \Delta = n!$$

together with a proof of correctness (z, c) which is computed as

$$v' = v^r, \tilde{x} = x^{4\Delta}, x' = \tilde{x}^r, c = H'(v, \tilde{x}, v_i, x_i^2, v', x'), z = s_i c + r$$

where $r \xleftarrow{\$} \mathbb{Z}_{2^{L(N)+2L_1}}$ and H' is another hash function whose output is an L_1-bit integer. The signature share would be correct if

$$c = H'(v, \tilde{x}, v_i, x_i^2, v^z v_i^{-c}, \tilde{x}^z x_i^{-2c})$$

which can be verified by the client [53].

Combining Shares. When the client collects valid shares from a set S of $t+1$ players, the signature can be produced as follows. It computes $w = \prod_{i \in S} x_i^{2\lambda_{0,i}^S}$ where

$$\lambda_{0,i}^S = \Delta \frac{\prod_{j \in S \setminus \{i\}} j}{\prod_{j \in S \setminus \{i\}} (j - i)} \in \mathbb{Z} \tag{2}$$

The factor Δ is needed to ensure that the coefficients are all integers. Once we have w, the signature y such that $y^e = x$ can be computed as

$$y = w^a x^b$$

where a, b are integers such that $e'a + eb = 1$ for $e' = 4\mathit{\Delta}^2$. a, b can be computed using extended Euclidean algorithm on e' and e. See [53].

Optimization. In practice, Shoup's scheme can be optimized to speedup the computation by moving some computation off-line. In particular, share combination uses Lagrange interpolation and needs to compute $\lambda_{0,i}^S$ with equation 2. Suppose the client chooses an initial set S of signers with $|S| = t + 1$. Equation 2 can be rewritten as

$$\lambda_{0,i}^S = \Delta \frac{\mathrm{num}_{0,i}^S}{\mathrm{den}_{0,i}^S}, \quad i \in S \tag{3}$$

The client pre-computes and stores $\lambda_{0,i}^S$ and den. Now suppose the client needs to combine shares from *another* set of signers S' with $|S'| = t + 1$ that differs from S by only a single element. Let $k \in S \setminus (S \cap S')$ and $k' \in S' \setminus (S \cap S')$. Then for $i \in S'$

$$\lambda_{0,i}^{S'} = \begin{cases} \lambda_{0,i}^S \cdot \frac{k'(k-i)}{k(k'-i)} & i \neq k', \\ \lambda_{0,i}^S \cdot \frac{\mathrm{den}_{0,i}^S}{\prod_{j \in S' \setminus \{i\}} (j-i)} & i = k'. \end{cases} \tag{4}$$

Computing all the $t + 1$ coefficients for S' using the original equation 2 requires $O(t^2)$ integer multiplications and divisions. Using the running update method, the cost is only $O(t)$. The saving could be significant (please see section 7 for empirical evaluation) since Δ is typically very large.

If S and S' differ by more than one elements, then the above process can be repeated multiple times.

Blinding. [11] uses blind signature to protect user data from the key server. We take a similar approach and present a threshold version. Blind signature is a variant of signature scheme that allows one to obtain a valid signature without leaking any information about the message to the signer. It was proposed by Chaum in [20] as a solution to obtaining anonymity in a digital cash system. The threshold version works as follows. To blind x, the requester generates a random group element $r \overset{\$}{\leftarrow} \mathbb{Z}_N^*$ and sends $\bar{x} \leftarrow r^e x \mod N$ to the signers. Combining $t + 1$ valid signature shares (on $\bar{x}$), the requester obtains $\bar{y}$ such that $\bar{x} = \bar{y}^{\,e} \mod N$. It then removes the blinding by $y \leftarrow \bar{y} r^{-1} \mod N$. y is then a standard RSA signature on M.

Clearly Shoup's threshold scheme can correctly recover y, as both x and $\bar{x}$ are elements in $\mathbb{Z}_N^*$, there is no difference for the signers to generate shares that combine into $\bar{y}$. The "unblinding" process works as a regular blind signature does [20].

To see this does not leak any information, we can use the same proof given in [20, 11]. Since $e > N$ is a prime, the map $f_e(x) = x^e \mod N$ is a permutation on $\mathbb{Z}_N^*$. Thus $r^e \mod N$ is a random element in $\mathbb{Z}_N^*$ for randomly chosen $r \in \mathbb{Z}_N^*$.

6.3 Security of DOPG-based EwS

We have established the deterministic D-IND$-CPA security of EwS. Our actual secure deduplication scheme replaces **Sign** with a distributed protocol introduced in the previous section. Using the simulation paradigm introduced in [43], we show that the adversary's view during the protocol can be simulated with a statistically close distribution by a polynomial time simulator who is given only access to the public key, corrupted users keys, and the deduplication information. From this, and the non-forgeability assumption of the threshold signature scheme, we prove that the protocol does not give the adversary more advantage in attacking the encryption.

To further empower the adversary, we allow the adversary to access more information than what it can obtain in an actual execution of our protocol. In addition to the ciphertexts and other transcript information, we also allow it to obtain the signature shares from honest users, for any chunk of its choice, up to a polynomial bound. We use a duplication oracle **Dup** to model the adversary's knowledge on duplication pattern as follows. We number the chunks $1, 2, \ldots, n$, where n is the maximum number of chunks the system could see, in the order they arrive at the system, and write c_i for the content of the i-th chunk. Duplication information for chunk i is the smallest integer j such that $j \leq i$ and $c_i = c_j$. That is **Dup** is defined as

$$\mathbf{Dup}(i) = \arg\min_j \text{ s.t. } j \leq i \text{ and } c_i = c_j$$

Before the game starts, the duplication oracle can be initialized with arbitrary duplication pattern. We require that the adversary's query messages must be consistent with the duplication oracle's equality pattern, meaning that, if $x_1, x_2, \ldots, x_n$ are the query messages for the signature shares and ciphertexts, in the order they are submitted, the following must hold:

$$x_i = x_j \quad \text{iff} \quad \mathbf{Dup}(i) = \mathbf{Dup}(j)$$

This is consistent with the condition in [10] and reflects the fact that every deterministic encryption leaks plaintext quality.

Theorem 4. *The view of the adversary can be simulated with a statistically close distribution by a polynomial time simulator who has access only to the adversary's keys and the duplication oracle (but not the signing key of the honest signers).*

Proof. The adversary's view consists of the signature shares from honest signers, the signatures, and the ciphertexts. The trick is how to simulate the signature shares and the signature without the signing keys. Fortunately there are standard techniques for doing that. Because the simulator controls H, we can simply select a random y and define $H(M) = y^e$ for any M. This makes y a standard RSA signature on message M with respect to H. There also exists a simulator for signature shares as constructed in [53]. Let S be such a simulator. Our simulator maintains a mapping `sigMap` from integers to numbers in $\mathbb{Z}_N^*$ and a counter `cnt` initialized to 1.

At the beginning of the game when the adversary chooses the corrupted signers, S is initialized with global keys and the IDs of the corrupted signers and their keys. Let Γ be the set of keys belonging to corrupted signers. S could simulate any honest signer's signature share on x given Γ and y such that $x = y^e$ [53]. When the adversary wants to see the signature share from signer i on chunk c, the simulator is implemented with the algorithm listed in figure 3.

```
$\mathbf{Sim}(i, c)$
$j \leftarrow \mathbf{Dup}(\texttt{cnt})$
If $j \neq i$ then
  $y \leftarrow \texttt{sigMap}[j]$
  If $y = \mathbf{null}$ then
    $y \leftarrow_R Z_N^*$
    $\texttt{sigMap}[j] \leftarrow y$
  Endif
Else
  $y \leftarrow_R Z_N^*$
  $\texttt{sigMap}[\texttt{cnt}] \leftarrow y$
Endif
$H[c] \leftarrow y^e \mod N$
$\texttt{cnt} \leftarrow \texttt{cnt} + 1$
If $G[y]$ is undefined then
    $G[y] \xleftarrow{\$} \{0,1\}^*$
Endif
$K \leftarrow G[y]$
$x_i \leftarrow S(\Gamma, y, H[c], i)$
Return $(x_i, \mathbf{SE}_K(c))$
```

Fig. 3. Simulator **Sim** for theorem 4

Essentially, on query (c, i), the simulator first queries the duplication oracle **Dup** with `cnt`. If **Dup** returns a number other than i, the simulator tries to retrieve $y \in Z_N^*$ from `sigMap`. If this is unsuccessful, it chooses a random $y \in Z_N^*$ and defines the output of H on c to be $y^e \mod N$. This ensures that y is a valid RSA signature with respect to hash H. It also adds to `sigMap` the mapping from `cnt` to y and increment `cnt`. The purpose of using `sigMap` and `cnt` is to ensure that the view is consistent with the deduplication pattern seen by the adversary.

The simulator passes $(\Gamma, y, H[c], i)$ to S which returns the corresponding signature share x_i. [6] it also feeds y to G to generate encryption key, denoted K. It then passes $(x_i, \mathbf{SE}_K(c))$ to the adversary. It is easy to verify that the simulator generates a view that is indistinguishable from a real run.

Theorem 4 implies that, EwS scheme, centralized or distributed, leaks no information, in an IND\$ sense, other than message equality. Since message equality is necessary for deduplication, this means that EwS achieves the best possible privacy under the current deduplication paradigm. Unless deduplication technique advances to a state that does not rely on message equality, we cannot hope for better.

6.4 Encrypted Deduplication

Using DOKG scheme described earlier, when the client needs to upload a chunk, c, it requests blind signature shares from $T \geq t+1$ signers. Once it receives enough shares, it performs share combination and unblinding to obtain the signature y on c. It then generates encryption key $K \leftarrow G(y)$ and encrypts c by $\mathbf{SE}_K(c)$. Its access to the data is ensured via the "lockbox" or key encapsulation mechanism [23, 49, 30]. A lockbox contains the encrypted keys: $lb(chunk) = \{\mathbf{AE}_{PK_i}(K) | i \in \Lambda\}$ where $\mathbf{AE}$ is an asymmetric encryption algorithm, K is the data encryption key, PK_i is the public key for user i, and Λ is the set of users having access right. [10] showed that such a scheme is IND-CPA secure. The lockbox is attached to the ciphertext and uploaded to the server.

A subtle issue is how a user could find its own key among the multiple slots in the lockbox. A simple solution is to explicitly store the user IDs. However, this allows the party having access to the lockbox, such as the server, to know who has access to a particular chunk. This may not be new information to the server but could be exploited by other parties (e.g., other users) if they somehow obtain the data. We thus recommend using the zero-knowledge lockbox scheme of [30] which uses a series of hash functions to locate a slot for a user. The scheme is quite efficient: the user could find her key in an average of $\log_2 m$ steps where m is the number of users who have access to this chunk. For detailed construction please see [30].

Once the ciphertext is uploaded, the deduplication process works essentially the same as what is being done in most existing storage services such as [28]: the server identify data redundancy by comparing hashes of ciphertexts and appending the lockboxes to the data to ensure access.

Group Access: In the case where the data should be made accessible to all the users in a group (but not to any party outside the group), there is no need to construct the lockboxes for each individual users. Instead, we can use the multicast encryption scheme of [31] and encrypt K using a multicast cryptosystem. The key structure of [31] is very similar to ours and it is straightforward to combine the two or, in the cases where the underlying threshold schemes (for

[6] Please see [53] for how S works.

multicast encryption and threshold signature) are the same, even let them share keys. For detailed construction please see [31].

7 Implementation and Evaluation

We have implemented our protocol and performed experiments to evaluate its feasibility. We adapted a Java implementation [7] of Shoup's threshold signature [53] with a number of optimizations to improve its performance. Our optimizations mainly include the following: (1) we replaced Java's built-in BigInteger with a NativeBigInteger implementation from the I2P anonymous network (http://www.i2p.net/); (2) we added a few pre-computations to move some work off-line, the most important one being the technique introduced in section 6.2. Compared with deduplication without encryption, the only change on the storage server's operation is that now the hash is computed over the ciphertext. There is no additional cost to the server's load. Therefore we only measured the impact on signers and the client.

Test Setting and Methodology. All tests were carried out with a 1024-bit RSA modulus with full-domain-hash using SHA256. All measurements on a single machine were repeated 1,000 times and all tests involving network were repeated 50 times. We report average running time for various benchmarks. Our protocol can be operated in two modes: semi-honest and malicious. In the semi-honest mode, the adversary is assumed to follow the protocol. In the malicious mode, on the other hand, the adversary is allowed to deviate arbitrarily from the protocol. The first mode represents a more benign environment, e.g., a well protected enterprise network where nodes are cooperative and not subject to external compromise. In this case we can omit the robustness mechanism, including the generation and verification of the proof of share correctness, for better performance. The second mode requires the full-fledged protocol in order to produce the correct signature. Note that the scheme is "fail-secure" in that if the protocol is set to run in the semi-honest mode but the adversary turns malicious, the result is incorrect signature, thus non-converging ciphertext and failed deduplication. As far as privacy is concerned, the protection is the same as semi-honest mode: unless the adversary corrupts $t+1$ players, the encryption is D-IND\$, assuming the underlying threshold signature scheme is non-forgeable and symmetric encryption is IND\$. This is in contrast to the DupLESS architecture where compromising a single key server reduces the security to that of CE.

Microbenchmarks. We first measured a few microbenchmarks on a 2.00GHz Intel(R) Xeon(R) E5-2650 running Linux. Table 1 shows the time needed to perform a number of basic operations for $n = 1024$ users.

Computation Time. Figure 4 shows, for both modes, the client computation time, in milliseconds, including combining signature shares, unblinding and producing the final encryption key, for varying t. To evaluate the effectiveness of

[7] http://sourceforge.net/projects/threshsig/

Table 1. Microbenchmarks (milliseconds)

Mode	KeyGen.	ShareGen.	ShareVer.
Semi-honest	880	0.7480	N/A
Malicious	9258	9.6432	4.5552

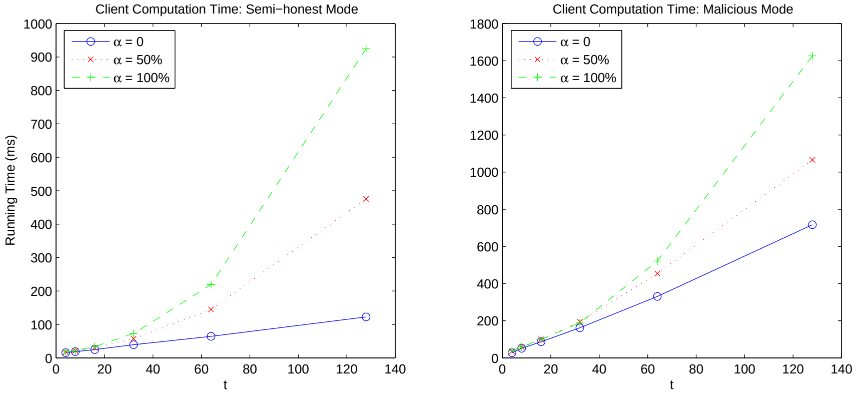

Fig. 4. Client computation time. Left: Semi-honest mode. Right: Malicious mode.

pre-computation introduced in section 6.2, we tested with different signer change rate α which is defined as follows. Let S be the original set of signers upon which $\lambda_{0,i}^{S}$'s are pre-computed and S' the actual set of signers whose shares are used to generate the signature, [8] α is defined as $\alpha = (|S'| - |S \cap S'|)/(t+1)$. $\alpha = 0$ means $S' = S$ and there is no need to update the pre-computed coefficients at all while $\alpha = 1$ means $S \cap S' = \emptyset$ and all the coefficients must be re-computed.

It is clear that the client's workload grows almost linearly with t, especially for full pre-computation settings (the lowest curves on both plots). The non-linearity is due to the cost for updating the Lagrange coefficients.

Complete Running Time. We then deployed the protocol in an actual network scenario. The signers and the clients are connected via a 1000Mbps LAN. This is a production cluster running routine jobs such as data mining MapReduce jobs. We run our tests during regular work hours, with normal work loads on the cluster.

Figure 5 shows, for both modes, the clients latency for the entire key generation process, including both computation (signer's signing time and client's combining time) and network latency.

Comparing figures 4 and 5, it is clear that computation time dominates. This should not be surprising since the signing time is quite small (0.7480 ms in semi-honest mode and 9.6432 ms in malicious mode) and our protocol only involves

[8] Due to the distributed nature, S and S' may not be identical. This could happen, e.g., when the client sends request to say $2(t+1)$ signers and uses the first $t+1$ responses to generate the signature.

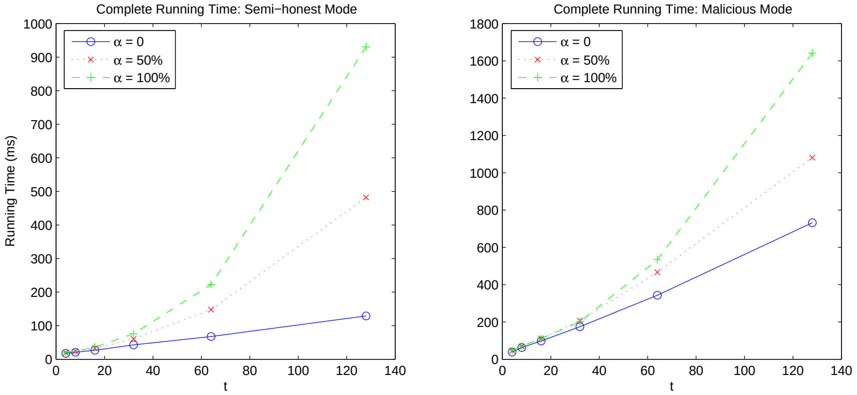

Fig. 5. Complete execution time. Left: Semi-honest mode. Right: Malicious mode.

very inexpensive communication: The user needs to receive $t+1$ signature shares together with proofs of correctness. All the shares are in the subgroup of squares in $\mathbb{Z}_N^*$. The data total no more than 100 KB even for $t = 128$.

At $t = 64$, which means the adversary must corrupt 65 signers to break the privacy, the entire latency for our key generation process is 533.26 milliseconds for the worst case (malicious mode and 100% signer change rate) and only 67.71 for the best case (semi-honest mode and 0 signer change rate).

Impact on Upload Throughput. We then used the numbers to evaluate the impact on client's upload throughput. let s be the user's original upload throughput without our scheme, measured in MB/s, c the chunk size in MB, and u the additional latency, in seconds, caused by our key generation process, then a user's throughput is

$$\frac{c}{c/s + u} = \frac{s}{1 + u\frac{s}{c}} = s\rho, \quad \text{where} \quad \rho = \frac{1}{1 + u\frac{s}{c}}$$

Figure 6 plots ρ versus varying values of c/s for a number of configurations for $t = 64$ which means that an adversary must corrupt 65 signers to break the security. It shows how the key generation process affects the uplink throughput for different choices of chunk sizes. The top curve is the best case scenario, with a semi-honest mode and full pre-computation, while the bottom curve represents the worst case where the protocol operates in malicious mode and the computation is fully online. If $c/s = 1$, which essentially means that originally it takes 1 second to upload the chunk, our scheme can retain about 90% of the original upload throughput in the best case and 70% in the worst. If $c/s = 2$, the ratios become about 96% and 80%, respectively. Clearly, the larger the c to s ratio is, the less relative overhead our scheme will cause on user's upload speed. From deduplication perspective, the choice of chunk size affects the effectiveness of deduplication. Smaller chunk size tends to offer stronger redundancy and more effective deduplication. However, small chunk size also introduces more overhead,

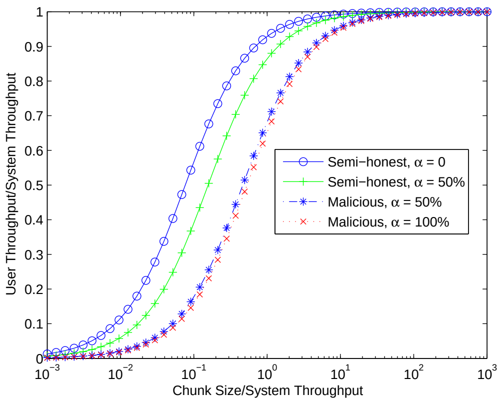

Fig. 6. Impact on user upload throughput for t = 64.

not only for our secure encryption, but also for deduplication. A good balance must be struck. [28] shows that, under typical workload of a corporate environment, a chunk size of 256KB could achieve about 35% space saving, while 32KB 42%. These chunk sizes are comparable to typical uplink speed of common users. This means that, using our scheme, we can choose a chunk size that is only about one or two times that of the amount of data a user could upload in one second and achieve substantial space saving via deduplication while retaining much of the original throughput *and* maintaining higher security. Using the benchmarks of DupLESS as references, uploading a 1 MB file with Dropbox takes 2700 milliseconds (ms) [11]. This translates into a system throughput of 0.37 MB/s. A chunk size of 256KB, achieving 35% space saving [28], results in $c/s = 0.68$, at which point our protocol retains about 90% of the original throughput in the best case, or 55% in the worst case, for $t = 64$. In addition, upload operations can often be made asynchronous to further reduce the impact on users' experience.

Fault Tolerance. In addition to performance and security benefits, a distributed scheme enjoys higher tolerance to faults. We did not run actual tests but a simple analysis shows that careful configuration can make the system highly robust in face of failures.

Let ϵ be the probability that a signer fails, meaning that it does not respond with a correct signature share during a certain period. Suppose for each chunk, the client sends signing requests to $T > t + 1$ signers. Assuming that the signers fail independently, then the probability that the client fails to obtain enough correct signature shares is

$$\sum_{i=0}^{t} \binom{T}{i} (1-\epsilon)^i \epsilon^{T-i} = \texttt{binocdf}(t, T, 1-\epsilon)$$

where `binocdf` is the binomial cumulative distribution function that can be found in many statistical analysis tools such as MATLAB. It is very easy to "reverse" it and compute the minimum T for given ϵ, t and target failure probability bound. Figure 7 shows the minimum T for varying t that bounds the clients failure probability due to signer failure by $\epsilon/1000$. Even for large t, and very severe node failure rate (40%), it suffices to use less than 300 signers for the scheme to be quite robust against signer failures.

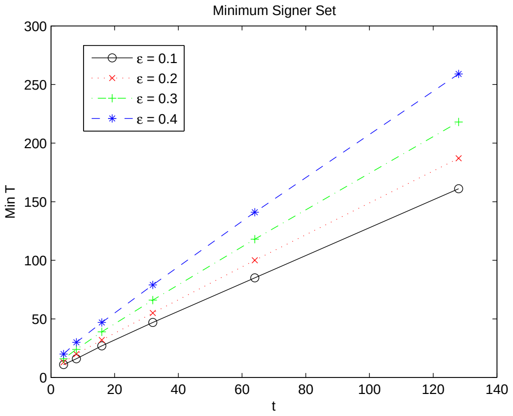

Fig. 7. Minimum number of signers to bound the client failure probability by $\epsilon/1000$ for a number of ϵs.

Our optimization is by no means thorough. A more aggressive optimization that utilizes more language and processor features could result in better performance. Our results, even in these sub-optimal cases, demonstrated that the scheme is practical for some reasonably large scale systems.

8 Conclusion and Discussion

In this paper we present a distributed key generation scheme supporting encrypted deduplication in cloud and other storage systems. We introduce a particularly strong notion of security for deterministic encryption and prove that our scheme and the generalized EwS are secure against such a notion. We show that the security level that our scheme achieves is the best under current deduplication paradigm. Our implementation demonstrates that the performance is adequate to support large-scale systems. In the following we discuss possible extensions to our scheme and future research directions.

8.1 Distributed Dealer

The scheme we described so far assumed a centralized dealer distributing the keys. It is actually possible to replace it with a distributed protocol so it can be made fully distributed. There are reasonably efficient distributed key generation protocols for sharing an RSA key (e.g., [19, 26, 40]). However, Shoup's threshold RSA that we use requires a special assumption on the RSA modulus: the modulus N must be the product of *safe* primes, i.e., $N = pq$ where $p' = (p-1)/2, q' = (q-1)/2$ are also primes. It is difficult to generate such a safe modulus in a distributed manner.

RSA modulus built with safe primes is needed in Shoup's scheme in two ways: [39] (1) for generating the proof of correctness of shares; and (2) for key generation to ensure that Shamir secret sharing is secure in the ring $\mathbb{Z}_{\phi(N)}$ where $\phi(N)$ is the Euler's totient function. Both are necessary if the dealer's role is to be distributed. There are two ways to work around it: (1) the key generation and distribution is still done by a trusted party (e.g., the users could find a trusted member to do it) but the signing process is distributed; or (2) use a less efficient, but still reasonably practical, variant of Shoup's scheme that allows for distributed key generation such as [39, 25]. We do not pursue this issue in this paper.

8.2 Measuring the Final Leakage: or Differential Privacy?

We have shown that, under current deduplication paradigm, full semantic security is impossible to achieve. In particular, the server learns the chunk deduplication pattern (e.g., the frequency that a chunk maps to other chunks). It remains open whether this is a serious leakage or not. This situation bears an interesting parallel to the challenge in statistical database privacy, which studies the problem of releasing statistical patterns of the data while preserving the privacy of each individual record [21, 45, 27, 14, 37, 36, 35, 50]. In 1977 Dalenius offered a semantic notion of privacy breach for statistical databases that is in the same spirit as semantic security in cryptography: "If the release of the statistic S makes it possible to determine the (microdata) value more accurately than without access to S, a disclosure has taken place" [24]. And it has also been proven that this is impossible to prevent if a database is to provide some degree of utility [34]. And the reason is simple: after all, one expects to learn some nontrivial facts from the data, which could be laws of nature (e.g., smoking causes lung cancer) or global statistics (e.g., average height) of a population. Both may allow an adversary to infer an individual's information [34].

Dwork [34] proposed the notion of *differential privacy* that is an achievable semantic notion based on *participation*: "we move from comparing an adversarys prior and posterior views of an individual to comparing the risk to an individual when included in, versus when not included in, the database." Formally it is defined as:

Definition 1 (Differential Privacy [37, 36]). $\forall \epsilon, \delta \geq 0$*, an algorithm* A *gives* (ϵ, δ)*-differential privacy if for all* $S \subseteq Range(A)$*, for all data sets* D, D' *such that* D *and* D' *differ by a single record*

$$\Pr[A(D) \in S] \leq \exp(\epsilon) \Pr[A(D') \in S] + \delta$$

A^f *is said to be* (ϵ, δ)*-private if it gives* (ϵ, δ)*-differential privacy.*

Differential privacy captures the intuition that the function is private if the risk to one's privacy does not substantially increase as a result of participating in the data set. It has been widely adopted by many works [16, 50, 48, 15, 47, 29, 32] and has became the "gold standard" of privacy in statistical database and privacy-preserving data mining.

It is therefore an interesting research question to ask: is differential privacy a meaningful notion for data privacy in the context of cloud storage, especially in the presence of deduplication? That is, can we also shift from measuring the difference between the adversarys prior and posterior views to comparing the risk to an individual when participating versus when not participating in the system? With our scheme, the only leakage is caused by the deduplication pattern, which can be seen as a utility that the server has to learn to perform its normal operations. It might be possible to cast it into a form that is compatible with differential privacy and examine whether the risk to one's privacy increases substantially as a result of storing one's data in the system. The mechanisms that achieve differential privacy (e.g., the ones based on additive noise [16, 50, 48, 15, 47] and those relying on aggregation [29, 32]) can then be used to guide the construction of a *private* deduplication algorithm that determines what and when to deduplicate to ensure that the *deduplication pattern* is differentially private. Unless there is a breakthrough in space saving techniques that does *not* need to exploit data redundancy, which appears to be unlikely in the foreseeable future, the above seems to be a viable approach that makes controllable compromise between privacy and utility. We leave this as a future research direction.

References

1. Bitcasa. http://www.bitcasa.com/.
2. Ciphertite. http://www.ciphertite.com.
3. Dropbox. http://www.dropbox.com/.
4. flud. http://flud.org.
5. Freenet. https://freenetproject.org/.
6. GNUnet. http://gnunet.org.
7. Anderson, P., and Zhang, L. Fast and secure laptop backups with encrypted de-duplication. In *Proceedings of the 24th international conference on Large installation system administration* (Berkeley, CA, USA, 2010), LISA'10, USENIX Association, pp. 1–8.
8. Bellare, M., Boldyreva, A., and O'Neill, A. Deterministic and efficiently searchable encryption. In *Proceedings of the 27th annual international cryptology conference on Advances in cryptology* (Berlin, Heidelberg, 2007), CRYPTO'07, Springer-Verlag, pp. 535–552. Full Version of this paper at http://www.cc.gatech.edu/ aboldyre/papers/bbo.pdf.
9. Bellare, M., Desai, A., Jokipii, E., and Rogaway, P. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1997), FOCS '97, IEEE Computer Society, pp. 394–403.
10. Bellare, M., Fischlin, M., O'Neill, A., and Ristenpart, T. Deterministic encryption: Definitional equivalences and constructions without random oracles. In

Proceedings of the 28th Annual conference on Cryptology: Advances in Cryptology (Berlin, Heidelberg, 2008), CRYPTO 2008, Springer-Verlag, pp. 360–378.

11. Bellare, M., and Keelveedhi, S. DupLESS: Server-aided encryption for deduplicated storage. In *USENIX Security Symposium 2013* (2013).
12. Bellare, M., Keelveedhi, S., and Ristenpart, T. Message-locked encryption and secure deduplication. In *Advances in Cryptology C EUROCRYPT 2013* (2013), T. Johansson and P. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 296–312.
13. Bennett, K., Grothoff, C., Horozov, T., and Patrascu, I. Efficient sharing of encrypted data. In *Proceedings of the 7th Australian Conference on Information Security and Privacy* (London, UK, UK, 2002), ACISP '02, Springer-Verlag, pp. 107–120.
14. Blum, A., Dwork, C., McSherry, F., and Nissim, K. Practical privacy: the SuLQ framework. In *PODS '05* (New York, NY, USA, 2005), ACM Press, pp. 128–138.
15. Blum, A., Ligett, K., and Roth, A. A learning theory approach to non-interactive database privacy. In *STOC 08* (2008), pp. 609–618.
16. Boaz Barak, e. a. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS '07* (New York, NY, USA, 2007), ACM Press, pp. 273–282.
17. Boldyreva, A., Fehr, S., and O'Neill, A. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Proceedings of the 28th Annual conference on Cryptology: Advances in Cryptology* (Berlin, Heidelberg, 2008), CRYPTO 2008, Springer-Verlag, pp. 335–359.
18. Bolosky, W. J., Douceur, J. R., Ely, D., and Theimer, M. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (New York, NY, USA, 2000), SIGMETRICS '00, ACM, pp. 34–43.
19. Boneh, D., and Franklin, M. Efficient generation of shared rsa keys. *J. ACM 48*, 4 (July 2001), 702–722.
20. Chaum, D. Blind signatures for untraceable payments. In *Advances in Cryptology*, D. Chaum, R. Rivest, and A. Sherman, Eds. Springer US, 1983, pp. 199–203.
21. Chin, F., and Ozsoyoglu, G. Auditing for secure statistical databases. In *ACM 81: Proceedings of the ACM '81 conference* (New York, NY, USA, 1981), ACM, pp. 53–59.
22. Clements, A. T., Ahmad, I., Vilayannur, M., and Li, J. Decentralized deduplication in san cluster file systems. In *Proceedings of the 2009 conference on USENIX Annual technical conference* (Berkeley, CA, USA, 2009), USENIX'09, USENIX Association, pp. 8–8.
23. Cramer, R., and Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology CRYPTO '98*, H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1998, pp. 13–25.
24. Dalenius, T. Towards a methodology for statistical disclosure control. *Statistik Tidskrift 15* (1977), 429–444.
25. Damgård, I., and Koprowski, M. Practical threshold rsa signatures without a trusted dealer. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology* (London, UK, UK, 2001), EUROCRYPT '01, Springer-Verlag, pp. 152–165.

26. Damgård, I., and Mikkelsen, G. L. Efficient, robust and constant-round distributed rsa key generation. In *Proceedings of the 7th international conference on Theory of Cryptography* (Berlin, Heidelberg, 2010), TCC'10, Springer-Verlag, pp. 183–200.
27. Dinur, I., and Nissim, K. Revealing information while preserving privacy. In *PODS '03* (New York, NY, USA, 2003), ACM Press, pp. 202–210.
28. Douceur, J. R., Adya, A., Bolosky, W. J., Simon, D., and Theimer, M. Reclaiming space from duplicate files in a serverless distributed file system. In *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)* (Washington, DC, USA, 2002), ICDCS '02, IEEE Computer Society, pp. 617–624.
29. Duan, Y. Privacy without noise. In *CIKM '09* (New York, NY, USA, 2009), ACM.
30. Duan, Y., and Canny, J. Protecting user data in ubiquitous computing: Towards trustworthy environments. In *PET'04* (2004).
31. Duan, Y., and Canny, J. How to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attack. In *RSA Conference 2006, Cryptographers' Track. San Jose, USA* (2006), vol. 3860 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 244–261.
32. Duan, Y., Canny, J., and Zhan, J. P4P: Practical large-scale privacy-preserving distributed computation robust against malicious users. In *USENIX Security Symposium 2010* (2010), pp. 609–618.
33. Dutch, M. Understanding data deduplication ratios.
34. Dwork, C. An ad omnia approach to defining and achieving private data analysis. In *PinKDD* (2007), pp. 1–13.
35. Dwork, C. Ask a better question, get a better answer a new approach to private data analysis. In *ICDT 2007* (2007), Springer, pp. 18–27.
36. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT 2006* (2006), Springer.
37. Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC 2006* (2006), Springer.
38. EMC. http://www.emc.com/solutions/samples/backup-recovery-archiving/backup-data-deduplication.htm.
39. Fouque, P.-A., and Stern, J. Fully distributed threshold RSA under standard assumptions. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology* (London, UK, UK, 2001), ASIACRYPT '01, Springer-Verlag, pp. 310–330.
40. Frankel, Y., MacKenzie, P. D., and Yung, M. Robust efficient distributed rsa-key generation. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 1998), PODC '98, ACM, pp. 320–.
41. Goldwasser, S., and Micali, S. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1982), STOC '82, ACM, pp. 365–377.
42. Goldwasser, S., and Micali, S. Probabilistic encryption. *Journal of Computer and System Sciences 28*, 2 (1984), 270 – 299.
43. Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1 (Feb. 1989), 186–208.

44. Goldwasser, S., Micali, S., and Rivest, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing 17*, 2 (1988), 281–308.

45. Kleinberg, J., Papadimitriou, C., and Raghavan, P. Auditing boolean attributes. In *PODS '00* (New York, NY, USA, 2000), ACM, pp. 86–91.

46. Langford, S. K. Threshold dss signatures without a trusted party. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1995), CRYPTO '95, Springer-Verlag, pp. 397–409.

47. McSherry, F., and Mironov, I. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD '09* (New York, NY, USA, 2009), ACM, pp. 627–636.

48. McSherry, F., and Talwar, K. Mechanism design via differential privacy. In *FOCS '07* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 94–103.

49. Miller, E. L., Long, D. D. E., Freeman, W. E., and Reed, B. Strong security for network-attached storage. In *Proceedings of the Conference on File and Storage Technologies* (Berkeley, CA, USA, 2002), FAST '02, USENIX Association, pp. 1–13.

50. Nissim, K., Raskhodnikova, S., and Smith, A. Smooth sensitivity and sampling in private data analysis. In *STOC '07* (2007), ACM, pp. 75–84.

51. Phan, D. H., and Pointcheval, D. Deterministic Symmetric Encryption (Semantic Security and Pseudo-Random Permutations). In *Proceedings of the 11th Annual Workshop on Selected Areas in Cryptography (SAC '04)* (Waterloo, Canada, 2004), H. Handschuh and M. A. Hasan, Eds., vol. 3357 of *Lecture Notes in Computer Science*, Springer, pp. 185–200.

52. Rogaway, P., Bellare, M., and Black, J. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur. 6*, 3 (Aug. 2003), 365–403.

53. Shoup, V. Practical threshold signatures. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 2000), EUROCRYPT'00, Springer-Verlag, pp. 207–220.

54. Wilcox-O'Hearn, Z. Convergent encryption reconsidered. https://tahoe-lafs.org/pipermail/tahoe-dev/2008-March/000449.html, 2008.

55. Wilcox-O'Hearn, Z., and Warner, B. Tahoe: the least-authority filesystem. In *Proceedings of the 4th ACM international workshop on Storage security and survivability* (New York, NY, USA, 2008), StorageSS '08, ACM, pp. 21–26.

56. Xing, Y., Li, Z., and Dai, Y. Peerdedupe: Insights into the peer-assisted sampling deduplication. In *Peer-to-Peer Computing* (2010), IEEE, pp. 1–10.

57. Zhao, X., Zhang, Y., Wu, Y., Chen, K., Jiang, J., and Li, K. Liquid: A scalable deduplication file system for virtual machine images. *IEEE Transactions on Parallel and Distributed Systems 99*, PrePrints (2013), 1.