# ROBUST AND PRIVATE DISTRIBUTED SHARED ATOMIC MEMORY IN MESSAGE PASSING NETWORKS

by

**Shlomi Dolev, Thomas Petig and Elad M. Schiller**

The Lynne and William Frankel Center for Computer Science
Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel

Technical Report #15-03

Feb. 2015

# Robust and Private Distributed Shared Atomic Memory in Message Passing Networks

(Submitted as a Brief Announcement to PODC'15)

Shlomi Dolev * Thomas Petig † Elad M. Schiller†

Security and privacy are often imperative for distributed systems. This motivates us to study the problem of emulating shared memory in message passing networks that include $N$ servers, at most $f$ crash-stop failures and $e$ semi-Byzantine servers that can deviate from the algorithm by sending corrupted data. We look at coded atomic storage algorithms that ensure no information leakage by letting the servers store their data as secret shares. We consider an enhancement of the coded atomic storage (CAS) algorithm by Cadambe et al. [3] in which we use $\lceil (N + k + 2e)/2 \rceil$-size quorums and Reed-Solomon codes.

The first algorithm for emulating a single-writer multi-reader shared memory by Attiya et al. [1], as well as the multi-writer multi-reader version by Fan and Lynch [4], handle link and node failures. Cadambe et al. [3] present the coded atomic storage (CAS) algorithm and improve communication and storage costs by using quorums and $(N, k)$-maximum distance separable (MDS) codes [7]. The CAS algorithm enables the reader to restore the data under the presence of $\frac{N-k}{2}$ stop-failed servers. We address privacy by storing on each node merely parts of the data, as in Shamir's secret sharing scheme [8], which we can implement using Reed-Solomon codes [5] and a matching error correction algorithm (Berlekamp-Welch [9]). This variation of the CAS algorithm also provides resilience against other errors, for example, data corruption of a bounded number of secret shares. We show how to combine shared-memory emulation with robustness and privacy.

## 1 Background

The $(N, k)$-*threshold scheme* is defined by Shamir [8] and splits a secret $s$ into $N$ secret shares $\{s_i\}_{i\in\{1,\ldots,N\}}$, so that there exists a mapping from $S \subseteq \{s_i\}_{i\in\{1,\ldots,N\}}$, to the secret $s$, but it is impossible to determine $s$ from a set of less than $k$ secret shares, where $k$ and $N$ are integers, such that $k < N$ and $k \leq |S|$. The $(N, k)$-Reed-Solomon code, $\Phi$, transforms the input data, i.e., one element of a $k$ dimensional vector space, $\mathcal{S}$, over a finite field $K$, into $N$ dimensional vector space, $\mathcal{W}$, over the same field, $K$. $k$ and $N$ are as above. We call $N$ the block length and $k$ the message length. The Berlekamp-Welch algorithm, $\Phi^{-1}$, can correct Reed-Solomon codes within $\mathcal{O}(N^3)$ time in the presence of $e$ errors and $f$ erasures, as long as $2e + f < N - k + 1$ [9]. Note that $(N, k)$-Reed-Solomon codes are a $(N, k)$-threshold scheme [5].

## 2 Our contribution

We show how to emulate atomic shared memory in the presence of semi-Byzantine servers. This approach ensures privacy. Namely, no group of up to $k-1$ servers is able to reconstruct the stored data, i.e., the secret.

*dolev@cs.bgu.ac.il, +972-86472718, Comp. Science, Ben-Gurion Univ. of the Negev, 84105 Beer-Sheva, Israel.

†{petig,elad}@chalmers.se, +46-317721053, Comp. Science, Chalmers Univ. of Tech., 41296 Gothenburg, Sweden.

Furthermore, a reader can reconstruct the correct secret even if up to $e$ servers deliver corrupted secret shares. We do that using Reed-Solomon codes [6] and the Berlekamp-Welch error correction algorithm [9]. This works because Cadambe et al. [3] use the class of the maximum-distance separable codes for the CAS algorithm, which includes the Reed-Solomon codes.

# 3 System Settings

We consider message passing networks in which nodes exchange messages via communication links. Messages are of the form $(t, w, d) \in \mathcal{T} \times \mathcal{W} \cup \{\perp\} \times \mathcal{D}$, where $\mathcal{T}$ is the set of tag tuples $(z, id)$ that contain an integer $z$ and a node identifier $id$. With $\mathcal{W}$ we denote, as mentioned, the set of secret shares, where $\perp$ is the invalid share and $\mathcal{D} := \{\text{'pre'}, \text{'fin'}\}$ is the label set. We distinguish among three node types: *server*, *reader* and *writer*. Each writer and each reader is reliably connected to all servers, and with bounded communication delays. Let $\mathcal{P}$ be the *server* set, where $N := |\mathcal{P}|$.

Our settings are motivated by (reliable) servers that stores large secret shares on (unreliable) mass storage systems. We allow at most $e$ semi-Byzantine servers and at most $f$ failures, such as communication delay violation. We assume that semi-Byzantine servers can send corrupted secret shares to readers, but not corrupted tags or labels, i.e., when a semi-Byzantine server replies with a tuple $(t, w, d)$, only $w$ might be corrupted. Writers split secrets using the $(N, k)$-Reed-Solomon code and submit the resulting secret shares to the servers. Servers store their secret shares and deliver them to the readers upon request. The proposed solution withstands a fault model that includes both server stop-failure and server semi-Byzantine behavior.

We say that a secret sharing protocol is $t$-*private* when a set of at most $t$ servers cannot compute the secret, as in [2]. Note that a 0-private protocol preserves no privacy. When the presence of at most $t$ semi-Byzantine servers and at most $s$ stop-failed servers does not influence the correctness of secret restored by a reader, we say that the protocol is $(s, t)$-*robust*, similar to $t$-*resilience* in [2].

# 4 Quorums of $(k + 2e)$-overlap

We define a *quorum* as a server subset $Q \subseteq \mathcal{P}$ with at least $\lceil \frac{N+k+2e}{2} \rceil$ elements, and we write $\mathcal{Q}$ as the all quorum set. Lemma 1 shows that any two different quorums share at least $k + 2e$ servers, rather than just $k$ of them as in Cadambe et al. [3]. These quorums guarantees that once a writer finishes its write operation, any reader can retrieve at least $k + 2e$ secret shares and reconstruct the secret.

**Lemma 1** (Variation of Cadambe et al. [3], Lemma 5.1)**.** *Suppose that* $1 \leq k \leq N - 2f - 2e$. ***(1)** If $Q_1, Q_2 \in \mathcal{Q}$, then $|Q_1 \cap Q_2| \geq k + 2e$. **(2)** The existence of such a $k$ implies the existence of $Q \in \mathcal{Q}$ such that $Q$ has no crashed servers.*

*Proof.* **(1)** Let $Q_1, Q_2 \in \mathcal{Q}$, then $|Q_1 \cap Q_2| = |Q_1| + |Q_2| - |Q_1 \cup Q_2| \geq 2 \left\lceil \frac{N+k+2e}{2} \right\rceil - N \geq k + 2e$. **(2)** Since there are at most $f$ crashed servers, we can show that without such $f$ servers, there are still enough alive servers for a quorum. It follows that $N - f \geq N - \left\lfloor \frac{N-k-2e}{2} \right\rfloor = \left\lceil \frac{N+k+2e}{2} \right\rceil$. □

By Lemma 1, the atomicity and liveness analysis in [3, Theorem 5.2 to Lemma 5.9] also holds when the CAS algorithm that uses $(k + 2e)$-overlap quorums.

# 5 The Algorithm

In order to tolerate at most $e$ (secret share corruptions made by) semi-Byzantine servers, we propose Algorithm 1 as a variation of Cadambe et al. [3] CAS algorithm that uses $(k + 2e)$-overlap quorums and

**Algorithm 1:** The robust and private coded atomic storage algorithm, code for $p_i$.

**Writer**: `// Writes secret s.`
Query for the highest finalized tag from a quorum, select the message $((z,k), w_{p_k}$, 'fin') such that $z$ is max.;
pre-write: Send $((z+1,i), \Phi_{p_j}(s)$, 'pre') to all $p_j \in \mathcal{P}$ and wait until quorum acknowledges;
finalize: Send $((z+1,i), \perp$, 'fin') to all $p_j \in \mathcal{P}$ and wait until quorum acknowledges;

**Reader**: `// Returns secret s, or ⊥ in case of failure.`
Query for the highest finalized tag from a quorum, select the tag $((z,j), w_{p_j}$, 'fin') such that $z$ is maximal;
Finalize: Send $(t, \perp$, 'fin') to all $s \in \mathcal{P}$ and let $Q$ be the response of a quorum ;
**if** $|\{(t, w_{p_j}$, *'fin'*$) \in Q : w_{p_j} \neq \perp\}| \geq k + 2e$ **then return** $\Phi^{-1}(\{(t, w_{p_j}, s) \in Q : w_{p_j} \neq \perp\})$ **else return** $\perp$;

**Server**: Storage variable: $S \subset \mathcal{T} \times (\mathcal{W} \cup \{\perp\}) \times \{$'pre', 'fin'$\}$;
**upon** (receive query) **do** Reply with highest finalized tag;
**upon** (receive pre-write $(t, w_i$, 'pre')) **do**
  **if** $\nexists (t, \bullet) \in S$ **then** $S \leftarrow S \cup (t, w_i$, 'pre');
  Reply with acknowledgement;
**upon** (receive finalize $(t, \perp$, 'fin') from writer) **do**
  **if** $\exists (t, w_i$, *'pre'*$) \in S$ **then** $S \leftarrow (S \setminus \{(t, w_i$, 'pre')$\}) \cup (t, w_i$, 'fin') **else** Add $(t, \perp$, 'fin') to $S$ ;
  Reply with acknowledgement;
**upon** (finalize $(t, \perp$, 'fin') from reader) **do**
  **if** $\exists (t, w_i, \bullet) \in S : w_i \neq \perp$ **then** $S \leftarrow (S \setminus \{(t, w_i, \bullet)\}) \cup \{(t, w_i$, 'fin')$\}$; reply $(t, w_i$, 'fin') ;
  **else** $S \leftarrow S \cup \{(t, \perp$, 'fin')$\}$; reply $(t, \perp$, 'fin') ;

$(N,k)$-Reed-Solomon codes [6], which is an $(N,k)$-MDS [7] code that Cadambe et al. [3] uses. By the atomicity and liveness analysis for the case of $(k+2e)$-overlap quorums (the remark after Lemma 1), the reader retrieves $k+2e$ unique secret shares that include at most $e$ manipulated shares.

**Corollary 1.** *Algorithm 1 emulates a shared atomic read/write memory if* $1 \leq k \leq N - 2f - 2e$.

# 6 Robustness

Robustness is added by the ability of the Berlekamp-Welch algorithm to correct error in the Reed-Solomon codes. Note that semi-Byzantine servers only introduce corrupted secret shares. Lemma 2 shows the robustness of Algorithm 1 against up to $f$ stop-failed and up to $e$ semi-Byzantine server.

**Lemma 2.** *For* $k \in \{1 \ldots, N - 2f - 2e\}$*, Algorithm 1* $(f,e)$*-robust.*

*Proof.* If a writer issues a query, pre-write and finalize operations it does not read back the secret from the server. Thus, writers are immune to semi-Byzantine server. Server do not exchange secrets with other server and thus are not directly affected by semi-Byzantine server. A reader collects secret shares from qourum of servers, but never writes them to servers, since a query and a finalize only contains a $\perp$ in place of a secret share. By Lemma 1 and Corollary 1 follows that a reader $p_i$ receives at least $k+2e$ secret shares from the finalize operation. From these $k+2e$ secret shares at most $e$ are corrupted and, thus, $p_i$ computes the correct secret by applying Berlekamp-Welch. □

# 7 Privacy

Our approach ensures privacy of the secret among servers. In Lemma 3 we see that a group of less than $k$ servers are not able to reconstruct the secret by combining the secret shares they have stored locally.

**Lemma 3.** *For* $1 \leq k \leq N - 2f - 2e$*, Algorithm 1 is* $(k-1)$*-private.*

*Proof.* Reed-Solomon codes implement the Shamir secret sharing [5] and, thus, $k-1$ servers cannot compute the secret using their $k-1$ local secret shares. □

Note that in the case $k = 1$, even if privacy is not protected, it is still possible to correct corrupted memory copies. This holds because the reader reads $1 + 2e$ secret shares and, thus, the additional $2e$ secret shares contain redundant information for the Berlekamp-Welch error correction.

# 8 Conclusions

Interestingly, fundamental building blocks for distributed systems can provide privacy and robustness. We show how to implement a robust and private coded atomic storage protocol, which is resilient to semi-Byzantine servers using shared memory emulation in message passing networks. In addition, our algorithm tolerates server crashes, and at the same time, it ensures the privacy of the stored data. We believe that our approach and techniques are useful for providing robustness and privacy for many more building blocks for distributed systems.

# 9 Bibliography


1. H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM (JACM)*, 42(1):124–142, 1995.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *20th Symp. on Theory of Comp.*, 1988, pages 1–10. ACM, 1988.
3. V. R. Cadambe, N. A. Lynch, M. Médard, and P. M. Musial. A coded shared atomic memory algorithm for message passing architectures. In *13th IEEE NCA*, pages 253–260. IEEE Computer Society, 2014.
4. R. Fan and N. A. Lynch. Efficient replication of large data objects. *17th DISC 2003*, vol. 2848 of *LNCS*, pages 75–91. Springer, 2003.
5. R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, Sept. 1981.
6. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2):300–304, 1960.
7. R. M. Roth. *Introduction to coding theory*. Cambridge University Press, 2006.
8. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
9. L. Welch and E. Berlekamp. Error correction for algebraic block codes, 1986/12/30. US Pat. 4,633,470.