Supplementary Material

A Algorithmic Illustration of LCC

Algorithm 1 LCC Encoding (Precomputation)

```
1: procedure Encode($X_1, X_2, ..., X_K, T$)                ▷ Encode inputs variables according to LCC
2:     generate uniform random variables $Z_{K+1}, ..., Z_{K+T}$
3:     jointly compute $\tilde{X}_i \leftarrow \sum_{j\in[K]} X_j \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{\alpha_i-\beta_k}{\beta_j-\beta_k} + \sum_{j=K+1}^{K+T} Z_j \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{\alpha_i-\beta_k}{\beta_j-\beta_k}$ for $i = 1, 2, ..., N$ using fast polynomial interpolation
4:     return $\tilde{X}_1, ..., \tilde{X}_N$                ▷ The coded variable assigned to worker $i$ is $\tilde{X}_i$
5: end procedure
```

Algorithm 2 Computation Stage

```
1: procedure WorkerComputation($\tilde{X}$)                ▷ Each worker $i$ takes $\tilde{X}_i$ as input
2:     return $f(\tilde{X})$                ▷ Compute as if no coding is taking place
3: end procedure
1: procedure Decode($S, A$)                ▷ Executed by master
2:     wait for a subset of fastest $N-S$ workers
3:     $\mathcal{N} \leftarrow$ identities of the fastest workers
4:     $\{\tilde{Y}_i\}_{i\in\mathcal{N}} \leftarrow$ results from the fastest workers
5:     recover $Y_1, ..., Y_K$ from $\{\tilde{Y}_i\}_{i\in\mathcal{N}}$ using fast interpolation or Reed-Solomon decoding   ▷ See Appendix B
6:     return $Y_1, ..., Y_K$
7: end procedure
```

$\beta_1, \ldots, \beta_{K+T}$ and $\alpha_1, ..., \alpha_N$ are global constants in $\mathbb{F}$, satisfying[18]

1. β_i's are distinct,
2. α_i's are distinct,
3. $\{\alpha_i\}_{i\in[N]} \cap \{\beta_j\}_{j\in[K]} = \varnothing$ (this requirement is alleviated if $T = 0$).

B Coding Complexities of LCC

By exploiting the algebraic structure of LCC, we can find efficient encoding and decoding algorithms with almost linear computational complexities. The encoding of LCC can be viewed as interpolating degree $K + T - 1$ polynomials, and then evaluating them at N points. It is known that both operations only require almost linear complexities: interpolating a polynomial of degree k has a complexity of $O(k \log^2 k \log\log k)$, and evaluating it at any k points requires the same [Kedlaya and Umans, 2011]. Hence, the total encoding complexity of LCC is at most $O(N \log^2(K+T) \log\log(K+T) \dim \mathbb{V})$, which is almost linear to the output size of the encoder $O(N \dim \mathbb{V})$.

Similarly, when no security requirement is imposed on the system (i.e., $A = 0$), the decoding of LCC can also be completed using polynomial interpolation and evaluation. An almost linear complexity $O(R \log^2 R \log\log R \dim \mathbb{U})$ can be achieved, where R denotes the recovery threshold.

A less trivial case is to consider the decoding algorithm when $A > 0$, where the goal is essentially to interpolate a polynomial with at most A erroneous input evaluations, or decoding a Reed-Solomon code. An almost linear time complexity can be achieved using additional techniques developed in [Berlekamp, 1968, Massey, 1969, Sudan, 1999, Rosenblum, 1999]. Specifically, the following $2A - 1$ *syndrome variables* can be computed with a complexity of $O((N-S) \log^2(N-S) \log\log(N-S) \dim \mathbb{U})$ using fast algorithms for polynomial evaluation and for transposed-Vandermonde-matrix multiplication [Pan, 2001].

$$S_k \triangleq \sum_{i\in\mathcal{N}} \frac{\tilde{Y}_i \alpha_i^k}{\prod_{j\in\mathcal{N}\setminus\{i\}}(\alpha_i - \alpha_j)} \qquad \forall k \in \{0, 1, ..., 2A-1\}. \tag{4}$$

[18]A variation of LCC is presented in Appendix D, by selecting different values of α_i's.

According to [Berlekamp, 1968, Massey, 1969], the location of the errors (i.e., the identities of adversaries in LCC decoding) can be determined given these syndrome variables by computing its rational function approximation. Almost linear time algorithms for this operation are provided in [Sudan, 1999, Rosenblum, 1999], which only requires a complexity of $O(A\log^2 A\log\log A\dim\mathbb{U})$. After identifying the adversaries, the final results can be computed similar to the $A=0$ case. This approach achieves a total decoding complexity of $O((N-S)\log^2(N-S)\log\log(N-S)\dim\mathbb{U})$, which is almost linear with respect to the input size of the decoder $O((N-S)\dim\mathbb{U})$.

Finally, note that the adversaries can only affect a *fixed* subset of A workers' results for all entries. This decoding time can be further reduced by computing the final outputs entry-wise: for each iteration, ignore computing results from adversaries identified in earlier steps, and proceed decoding with the rest of the results.

C The MDS property of U^{bottom}

Lemma 2. *The matrix U^{bottom} is an MDS matrix.*

Proof. First, let $V\in\mathbb{F}^{T\times N}$ be

$$V_{i,j}=\prod_{\ell\in[T]\setminus\{i\}}\frac{\alpha_j-\beta_{\ell+K}}{\beta_{i+K}-\beta_{\ell+K}}.$$

It follows from the resiliency property of LCC that by having $(\tilde{X}_1,\ldots,\tilde{X}_N)=(X_1,\ldots,X_T)\cdot V$, the master can obtain the values of $X_1,\ldots,X_T$ from any T of the $\tilde{X}_i$'s. This is one of the alternative definitions for an MDS code, and hence, V is an MDS matrix.

To show that U^{bottom} is an MDS matrix, it is shown that U^{bottom} can be obtained from V by multiplying rows and columns by nonzero scalars. Let $[T:K]\triangleq\{T+1,T+2,\ldots,T+K\}$, and notice that for $(s,r)\in[T]\times[N]$, entry (s,r) of U^{bottom} can be written as

$$\prod_{t\in[K+T]\setminus\{s+K\}}\frac{\alpha_r-\beta_t}{\beta_{s+K}-\beta_t}=\prod_{t\in[K]}\frac{\alpha_r-\beta_t}{\beta_{s+K}-\beta_t}\cdot\prod_{t\in[K:T]\setminus\{s+K\}}\frac{\alpha_r-\beta_t}{\beta_{s+K}-\beta_t}.$$

Hence, U^{bottom} can be written as

$$U^{bottom}=\operatorname{diag}\left(\left(\prod_{t\in[K]}\frac{1}{\beta_{s+K}-\beta_t}\right)_{s\in[T]}\right)\cdot V\cdot \operatorname{diag}\left(\left(\prod_{t\in[K]}(\alpha_r-\beta_t)\right)_{r\in[N]}\right),\tag{5}$$

where V is a $T\times N$ matrix such that

$$V_{i,j}=\prod_{t\in[T]\setminus\{i\}}\frac{\alpha_j-\beta_{t+K}}{\beta_{i+K}-\beta_{t+K}}.$$

Since $\{\beta_t\}_{t=1}^K\cap\{\alpha_r\}_{r=1}^N=\varnothing$, and since all the β_i's are distinct, it follows from (5) that U^{bottom} can be obtained from V by multiplying each row and each column by a nonzero element, and hence U^{bottom} is an MDS matrix as well. □

D The Uncoded Version of LCC

In Section 4.2, we have described the LCC scheme, which provides an S-resilient, A-secure, and T-private scheme as long as $(K+T-1)\deg f+S+2A+1\le N$. Instead of explicitly following the same construction, a variation of LCC can be made by instead selecting the values of α_i's from the set $\{\beta_j\}_{j\in[K]}$ (not necessarily distinctly).

We refer to this approach as the *uncoded version of LCC*, which essentially recovers the *uncoded repetition* scheme, which simply replicates each X_i onto multiple workers. By replicating every X_i between $\lfloor N/K \rfloor$ and $\lceil N/K \rceil$ times, it can tolerate at most S stragglers and A adversaries, whenever

$$S + 2A \leq \lfloor N/K \rfloor - 1, \tag{6}$$

which achieves the optimum resiliency and security when the number of workers is small and no data privacy is required (specifically, $N < K \deg f - 1$ and $T = 0$, see Section 5).

When privacy is taken into account (i.e., $T > 0$), an alternative approach in place of repetition is to instead store each input variable using Shamir's secret sharing scheme [Shamir, 1979] over $\lfloor N/K \rfloor$ to $\lceil N/K \rceil$ machines. This approach achieves any (S, A, T) tuple whenever $N \geq K(S + 2A + \deg f \cdot T + 1)$. However, it does not improve LCC.

E Proof of Lemma 1

We start by defining the following notations. For any multilinear function f defined on $\mathbb{V}$ with degree d, let $X_{i,1}, X_{i,2}, ..., X_{i,d}$ denote its d input entries (i.e., $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,d})$ and f is linear with respect to each entry). Let $\mathbb{V}_1, ..., \mathbb{V}_d$ be the vector space that contains the values of the entries. For brevity, we denote $\deg f$ by d in this appendix. We first provide the proof of inequality (3).

Proof of inequality (3). Without loss of generality, we assume both the encoding and decoding functions are deterministic in this proof, as the randomness does not help with decodability.[19] Similar to [Yu et al., 2018], we define the minimum recovery threshold, denoted by $R^*(N, K, f)$, as the minimum number of workers that the master has to wait to guarantee decodability, among all linear encoding schemes. Then we essentially need to prove that $R^*(N, K, f) \geq R^*_{\text{LCC}}(N, K, f)$, i.e., $R^*(N, K, f) \geq (K-1)d + 1$ when $N \geq Kd - 1$, and $R^*(N, K, f) \geq N - \lfloor N/K \rfloor + 1$ when $N < Kd - 1$.

Obviously $R^*(N, K, f)$ is a non-decreasing function with respect to N. Hence, it suffices to prove that $R^*(N, K, f) \geq N - \lfloor N/K \rfloor + 1$ when $N \leq Kd - 1$. We prove this converse bound by induction.

(a) If $d = 1$, then f is a linear function, and we aim to prove $R^*(N, K, f) \geq N + 1$ for $N \leq K - 1$. This essentially means that no valid computing schemes can be found when $N < K$. Assuming the opposite, suppose we can find a valid computation design using at most $K - 1$ workers, then there is a decoding function that computes all $f(X_i)$'s given the results from these workers.

Because the encoding functions are linear, we can thus find a non-zero vector $(a_1, ..., a_K) \in \mathbb{F}^K$ such that when $X_i = a_i V$ for any $V \in \mathbb{V}$, the coded variable $\tilde{X}_i$ stored by any worker equals the padded random key, which is a constant. This leads to a fixed output from the decoder. On the other hand, because f is assumed to be non-zero, the computing results $\{f(X_i)\}_{i \in [K]}$ is variable for different values of V, which leads to a contradiction. Hence, we have prove the converse bound for $d = 1$.

(b) Suppose we have a matching converse for any multilinear function with $d = d_0$. We now prove the lower bound for any multilinear function f of degree $d_0 + 1$. Similar to part (a), it is easy to prove that $R^*(N, K, f) \geq N + 1$ for $N \leq K - 1$. Hence, we focus on $N \geq K$.

The proof idea is to construct a multilinear function f' with degree d_0 based on function f, and to lower bound the minimum recovery threshold of f using that of f'. More specifically, this is done by showing that given any computation design for function f, a computation design can also be developed for the corresponding f', which achieves a recovery threshold that is related to that of the scheme for f.

In particular, for any non-zero function $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0+1})$, we let f' be a function which takes inputs $X_{i,1}, X_{i,2}, ..., X_{i,d_0}$ and returns a linear map, such that given any $X_{i,1}, X_{i,2}, ..., X_{i,d_0+1}$, we have $f'(X_{i,1}, X_{i,2}, ..., X_{i,d_0})(X_{i,d_0+1}) = f(X_{i,1}, X_{i,2}, ..., X_{i,d_0+1})$. One can verify that f' is a multilinear function with degree d_0, Given parameters K and N, we now develop a computation strategy for f' for a dataset of K inputs and a cluster of $N' \triangleq N - K$ workers, which achieves a recovery threshold of $R^*(N, K, f) - (K - 1)$. We construct this computation strategy based on an encoding strategy of f that achieves the recovery threshold $R^*(N, K, f)$. For brevity, we refer to these two schemes as the f'-scheme and f-scheme respectively.

[19]Note that this argument requires the assumption that the decoder does not have access to the random keys, as assumed in Section 2.

Because the encoding functions are linear, we consider the encoding matrix, denoted by $G \in \mathbb{F}^{K\times N}$, and defined as the coefficients of the encoding functions $\tilde{X}_i = \sum_{j=1}^{K} X_j G_{ji} + \tilde{z}_i$, where $\tilde{z}_i$ denotes the value of the random key padded to variable $\tilde{X}_i$. Following the same arguments we used in the $d = 1$ case, the left null space of G must be $\{0\}$. Consequently, the rank of G equals K, and we can find a subset $\mathcal{K}$ of K workers such that the corresponding columns of G form a basis of $\mathbb{F}^K$. Hence, we can construct the f'-scheme by letting each of the $N' \triangleq N - K$ workers store the coded version of $(X_{i,1}, X_{i,2}, \ldots, X_{i,d_0})$ that is stored by a unique respective worker in $[N] \setminus \mathcal{K}$ in f-scheme.[20]

Now it suffices to prove that the above construction achieves a recovery threshold of $R^*(N, K, f) - (K - 1)$. Equivalently, we need to prove that given any subset $\mathcal{S}$ of $[N]\backslash\mathcal{K}$ of size $R^*(N, K, f) - (K - 1)$, the values of $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x)$ for any $i \in [K]$ and $x \in \mathbb{V}$ are decodable from the computing results of workers in $\mathcal{S}$.

We exploit the decodability of the computation design for function f. For any $j \in \mathcal{K}$, the set $\mathcal{S} \cup \mathcal{K}\backslash\{j\}$ has size $R^*(N, K, f)$. Consequently, for any vector $(x_{1,d_0+1}, ..., x_{K,d_0+1}) \in \mathbb{V}^K_{d_0+1}$, we have that $\{f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x_{i,d_0+1})\}_{i\in[K]}$ is decodable given the results from workers in $\mathcal{S} \cup \mathcal{K}\backslash\{j\}$ computed in f-scheme, if each x_{i,d_0+1} is used as the $(d_0 + 1)$th entree for each input.

Because columns of G with indices in $\mathcal{K}$ form a basis of $\mathbb{F}^K$, we can find values for each input X_{i,d_0+1} such that workers in $\mathcal{K}$ would store 0 for the X_{i,d_0+1} entry in the f-scheme. We denote these values by $\bar{x}_{1,d_0+1}, ..., \bar{x}_{K,d_0+1}$. Note that if these values are taken as inputs, workers in $\mathcal{K}$ would return constant 0 due to the multilinearity of f. Hence, decoding $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, \bar{x}_{i,d_0+1})$ only requires results from workers not in $\mathcal{K}$, i.e., it can be decoded given computing results from workers in $\mathcal{S}$ using the f-scheme. Note that these results can be directly computed from corresponding results in the f'-scheme. We have proved the decodability of $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x)$ for $x = \bar{x}_{i,d_0+1}$.

Now it remains to prove the decodability of $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x)$ for each i for general $x \in \mathbb{V}$. For any $j \in \mathcal{K}$, let $\boldsymbol{a}^{(j)} \in \mathbb{F}^K$ be a non-zero vector that is orthogonal to all columns of G with indices in $\mathcal{K}\backslash\{j\}$. If $a_i^{(j)} x + \bar{x}_{i,d_0+1}$ is used for each input X_{i,d_0+1} in the f-scheme, then workers in $\mathcal{K}\backslash\{j\}$ would store 0 for the X_{i,d_0+1} entry, and return constant 0 due to the multilinearity of f. Recall that $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, a_i^{(j)} x + \bar{x}_{i,d_0+1})$ is assumed to be decodable in the f-scheme given results from workers in $\mathcal{S} \cup \mathcal{K}\backslash\{j\}$. Following the same arguments above, one can prove that $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, a_i^{(j)} x + \bar{x}_{i,d_0+1})$ is also decodable using the f'-scheme. Hence, the same applies for $a_i^{(j)} f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x)$ due to multilinearity of f.

Because columns of G with indices in $\mathcal{K}$ form a basis of $\mathbb{F}^K$, the vectors $\boldsymbol{a}^{(j)}$ for $j \in \mathcal{K}$ also from a basis. Consequently, for any i there is a non-zero $a_i^{(j)}$, and thus $f(X_{i,1}, X_{i,2}, ..., X_{i,d_0}, x)$ is decodable. This completes the proof of decodability.

To summarize, we have essentially proved that $R^*(N, K, f) - (K - 1) \geq R^*(N - K, K, f')$. We can verify that the converse bound $R^*(N, K, f) \geq N - \lfloor N/K \rfloor + 1$ under the condition $N \leq Kd - 1$ can be derived given the above result and the induction assumption, for any function f with degree $d_0 + 1$.

(c) Thus, a matching converse holds for any $d \in \mathbb{N}_+$, which proves inequality (3). □

Now we proceed to prove the rest of Lemma 1, explicitly, we aim to prove that the recovery threshold of any T-private encoding scheme is at least $R_{\text{LCC}}(N, K, f) + T \cdot \deg f$. Inequality (3) essentially covers the case for $T = 0$. Hence, we focus on $T > 0$. To simplify the proof, we prove a stronger version of this statement: when $T > 0$, any valid T-private encoding scheme uses at least $N \geq R_{\text{LCC}}(N, K, f) + T \cdot \deg f$ workers. Equivalently, we aim to show that $N \geq (K + T - 1) \deg f + 1$ for any such scheme.

We prove this fact using an inductive approach. To enable an inductive structure, we prove a even stronger converse by considering a more general class of computing tasks and a larger class of encoding schemes, formally stated in the following lemma.

Lemma 3. *Consider a dataset with inputs $X \triangleq (X_1, ..., X_K) \in (\mathbb{F}^d)^K$, and an input vector $\Gamma \triangleq (\Gamma_1, ..., \Gamma_K)$ which belongs to a given subspace of $\mathbb{F}^K$ with dimension $r > 0$; a set of N workers where each can take a coded variable in $\mathbb{F}^{d+1}$ and return the product of its elements; and a computing task where the master aim to recover $Y_i \triangleq X_{i,1} \cdot ... \cdot X_{i,d} \cdot \Gamma_i$. If the inputs entries are encoded separately such that each of the first d entries assigned*

[20]For breivity, in this proof we instead index these $N - K$ workers also using the set $[N] \setminus \mathcal{K}$, following the natural bijection.

to each worker are some $T_{\mathrm{X}} > 0$-privately linearly coded version of the corresponding entries of X_i's, and the $(d+1)$th entry assigned to each worker is a T-privately[21] *linearly coded version of Γ, moreover, if each Γ_i (as a variable) is non-zero, then any valid computing scheme requires $N \geq (T_{\mathrm{X}} + K - 1)d + T + r$.*

Proof. Lemma 3 is proved by induction with respect to the tuple (d, T, r). Specifically, we prove that (a) Lemma 3 holds when $(d, T, r) = (0, 0, 1)$; (b) If Lemma 3 holds for any $(d, T, r) = (d_0, 0, r_0)$, then it holds when $(d, T, r) = (d_0, 0, r_0 + 1)$; (c) If Lemma 3 holds for any $(d, T, r) = (d_0, 0, r_0)$, then it holds when $(d, T, r) = (d_0, T, r_0)$ for any T; (d) If Lemma 3 holds for any $d = d_0$ and arbitrary values of T and r, then it holds if $(d, T, r) = (d_0 + 1, 0, 1)$. Assuming the correctness of these statements, Lemma 3 directly follows by induction's principle. Now we provide the proof of these statements as follows.

(a). When $(d, T, r) = (0, 0, 1)$, we need to show that at least 1 worker is needed. This directly follows from the decodability requirement, because the master aims to recover a variable, and at least one variable is needed to provide the information.

(b). Assuming that for any $(d, T, r) = (d_0, 0, r_0)$ and any K and T_{X}, any valid computing scheme requires $N \geq (T_{\mathrm{X}} + K - 1)d_0 + r$ workers, we need to prove that for $(d, T, r) = (d_0, 0, r_0 + 1)$, at least $(T_{\mathrm{X}} + K - 1)d_0 + r_0 + 1$ workers are needed. We prove this fact by fixing an arbitrary valid computing scheme for $(d, T, r) = (d_0, 0, r_0 + 1)$. For brevity, let $\tilde{\Gamma}_i$ denotes the coded version of Γ stored at worker i. We consider the following two possible scenarios: (i) there is a worker i such that $\tilde{\Gamma}_i$ is not identical (up to a constant factor) to any variable Γ_j, or (ii) for any worker i, $\tilde{\Gamma}_i$ is identical (up to a constant factor) to some Γ_j.

For case (i), similar to the ideas we used to prove inequality (3), it suffices to show that if the given computing scheme uses N workers, we can construct another computation scheme achieving the same T_X, for a different computing task with parameters $d = d_0$ and $r = r_0$, using at most $N - 1$ workers.

Recall that we assumed that there is a worker i, such that $\tilde{\Gamma}_i$ is not identical (up to a constant factor) to any Γ_j. We can always restrict the value of Γ to a subspace with dimension r_0, such that $\tilde{\Gamma}_i$ becomes a constant 0. After this operation, from the computation results of the rest $N - 1$ workers, the master can recover a computing function with $r = r_0$ and non-zero Γ_j's, which provides the needed computing scheme.

For case (ii), because each Γ_j is assumed to be non-zero, we can partition the set of indices j into distinct subsets, such that any j and j' are in the same subset iff Γ_j is a constant multiple of $\Gamma_{j'}$. We denote these subsets by $\mathcal{J}_1, ..., \mathcal{J}_m$. Moreover, for any $k \in [m]$, let $\mathcal{I}_k$ denote the subset of indices i such that $\tilde{\Gamma}_i$ is identical (up to a constant factor) to Γ_j for j in $\mathcal{J}_k$.

Now for any $k \in [m]$, we can restrict the value of Γ to a subspace with dimension r_0, such that Γ_j is zero for any $j \in \mathcal{J}_k$. After applying this operation, from the computation results of workers in $[N]\backslash\mathcal{I}_k$, the master can recover a computing function with $r = r_0$, where $K' = K - |\mathcal{J}_k|$ sub-functions has non-zero Γ_j's. By applying the induction assumption on this provided computing scheme, we have $N - |\mathcal{I}_k| \geq (T_{\mathrm{X}} + K - |\mathcal{J}_k| - 1)d_0 + r_0$. By taking the summation of the this inequality over $k \in [m]$, we have

$$Nm - \sum_{k=1}^{m} |\mathcal{I}_k| \geq (T_{\mathrm{X}}m + Km - K - m)d_0 + r_0 m. \tag{7}$$

Recall that for any worker i, $\tilde{\Gamma}_i$ is identical (up to a constant factor) to some Γ_j, we have $\cup_{k\in[m]}\mathcal{I}_k = [N]$. Thus, $\sum_k |\mathcal{I}_k| \geq N$. Consequently, inequality (7) implies that

$$Nm - N \geq (T_{\mathrm{X}}m + Km - K - m)d_0 + r_0 m. \tag{8}$$

Note that $r_0 + 1 > 1$, which implies that at least two Γ_j's are not identical up to a constant factor. Hence, $m - 1 > 0$, and (8) is equivalently

$$N \geq \frac{(T_{\mathrm{X}}m + Km - K - m)d_0 + r_0 m}{m - 1} \tag{9}$$

$$= (T_{\mathrm{X}} + K - 1)d_0 + r_0 + ((T_{\mathrm{X}} - 1)d_0 + r_0)\,\frac{1}{m-1}. \tag{10}$$

[21] For this lemma, we assume that no padded random variable is used for a 0-private encoding scheme.

Since $T_{\rm X}$ and r_0 are both positive, we have $(T_{\rm X}-1)d_0+r_0>0$. Consequently, $((T_{\rm X}-1)d_0+r_0)\,\frac{1}{m-1}>0$, and we have

$$N \geq (T_{\rm X}+K-1)d_0+r_0+1, \tag{11}$$

which proves the induction statement.

(c). Assuming that for any $(d,T,r)=(d_0,0,r_0)$, any valid computing scheme requires $N\geq(T_{\rm X}+K-1)d_0+r_0$ workers, we need to prove that for $(d,T,r)=(d_0,T_0,r_0)$, $N\geq(T_{\rm X}+K-1)d_0+T_0+r_0$. Equivalently, we aim to show that for any $T_0>0$, in order to provide T_0-privacy to the d_0+1th entry, T_0 extra worker is needed. Similar to the earlier steps, we consider an arbitrary valid computing scheme for $(d,T,r)=(d_0,T_0,r_0)$ that uses N workers. We aim to construct a new scheme for $(d,T,r)=(d_0,0,r_0)$, for the same computation task and the same $T_{\rm X}$, which uses at most $N-T_0$ workers.

Recall that if an encoding scheme is T_0 private, then given any subset of at most T_0 workers, denoted by $\mathcal{T}$, we have $I(\Gamma;\tilde{\Gamma}_{\mathcal{T}})=0$. Consequently, conditioned on $\tilde{\Gamma}_{\mathcal{T}}=0$, the entropy of the variable Γ remains unchanged. This indicates that Γ can be any possible value when $\tilde{\Gamma}_{\mathcal{T}}=0$. Hence, we can let the values of the padded random variables be some linear combinations of the elements of Γ, such that worker in $\mathcal{T}$ returns constant 0.

Now we construct an encoding scheme as follows. Firstly it is easy to show that when the master aims to recover a non-constant function, at least T_0+1 workers are needed to provide non-zero information regarding the inputs. Hence, we can arbitrarily select a subset of T_0 workers, denoted by $\mathcal{T}$. As we have proved, we can find fix the values of the padded random variables such that $\tilde{\Gamma}_{\mathcal{T}}=0$. Due to multilinearity of the computing task, these workers in $\mathcal{T}$ also returns constant 0. Conditioned on these values, the decoder essentially computes the final output only based on the rest $N-T_0$ workers, which provides the needed computing scheme. Moreover, as we have proved that the values of the padded random variables can be chosen to be some linear combinations of the elements of Γ, our obtained computing scheme encodes Γ linearly. This completes the proof for the induction statement.

(d). Assuming that for any $d=d_0$ and arbitrary values of T and r, any valid computing scheme requires $N\geq(T_{\rm X}+K-1)d_0+T+r$ workers, we need to prove that for $(d,T,r)=(d_0+1,0,1)$, $N\geq(T_{\rm X}+K-1)(d_0+1)+1$. Observing that for any computing task with $r=1$, by fixing an non-zero Γ, it essentially computes K functions where each multiplies d_0 variables. Moreover, for each function, by viewing the first (d_0-1) entries as a vector X'_i and by viewing the last entry as a scalar Γ'_i, it essentially recovers the case where the parameter d is reduced by 1, K remain unchanged, and r equals K. By adapting any computing scheme in the same way, we have $T_{\rm X}$ remain unchanged, and T becomes $T_{\rm X}$. Then by induction assumption, any computing scheme for $(d,T,r)=(d_0+1,0,1)$ requires at least $(T_{\rm X}+K-1)d_0+T_{\rm X}+K=(T_{\rm X}+K-1)(d_0+1)+1$ workers. □

Remark 6. Using exactly the same arguments, Lemma 3 can be extended to the case where the entries of X are encoded under different privacy requirements. Specifically, if the ith entry is T_i-privately encoded, then at least $\sum_{i=1}^{d}T_i+(K-1)d+T+r$ worker is needed. Lemma 3 and this extended version are both tight, in the sense for any parameter values of d, K and r, there are computing tasks where a computing scheme that uses the matching number of workers can be found, using constructions similar to the Lagrange coded computing.

Now using Lemma 3, we complete the proof of Lemma 1 for $T>0$. Similar to the proof ideas for inequality (3) part (a), we consider any multilinear function f with degree d, and we find constant vectors $V_1,...,V_d$, such that $f(V_1,...,V_d)$ is non-zero. Then by restricting the input variables to be constant multiples of $V_1,...,V_d$, this computing task reduces to multiplying d scalars, given K inputs. As stated in Lemma 3 and discussed in part (d) of its induction proof, such computation requires $(T+K-1)d+1$ workers. This completes the proof of Lemma 1.

F Optimality on the Resiliency-Security-Privacy Tradeoff for Multilinear Functions

In this appendix, we prove the first part of Theorem 2 using Lemma 1. Specifically, we aim to prove that LCC achieves the optimal trade-off between resiliency, security, and privacy for any multilinear function f. By comparing Lemma 1 and the achievability result presented in Theorem 1 and Appendix D, we essentially need to show that for any linear encoding scheme that can tolerates A adversaries and S stragglers, it can also tolerate $S+2A$ stragglers.

This converse can be proved by connecting the straggler mitigation problem and the adversary tolerance problem using the extended concept of Hamming distance for coded computing, which is defined in [Yu et al., 2018].

Specifically, given any (possibly random) encoding scheme, its hamming distance is defined as the minimum integer, denoted by d, such that for any two instances of input X whose outputs Y are different, and for any two possible realizations of the N encoding functions, the computing results given the encoded version of these two inputs, using the two lists of encoding functions respectively, differs for at least d workers.

It was shown in [Yu et al., 2018] that this hamming distance behaves similar to its classical counter part: an encoding scheme is S-resilient and A-secure whenever $S + 2A \leq d - 1$. Hence, for any encoding scheme that is A-secure and S-reselient, it has a hamming distance of at least $S + 2A + 1$. Consequently it can tolerate $S + 2A$ stragglers. Combining the above and Lemma 1, we have completed the proof.

G Optimality on the Resiliency-Privacy Tradeoff for General Multivariate Polynomials

In this appendix, we prove the second part of Theorem 2 using Lemma 1. Specifically, we aim to prove that LCC achieves the optimal trade-off between resiliency and privacy, for general multivariate polynomial f. The proof is carried out by showing that for any function f that allows S-resilient T-private designs, there exists a multilinear function with the same degree for which a computation scheme can be found that achieves the same requirement.

Specifically, given any function f with degree d, we aim to provide an explicit construction of an multilinear function, denoted by f', which achieves the same requirements. The construction satisfies certain properties to ensure this fact. Both the construction and the properties are formally stated in the following lemma (which is proved in Appendix H):

Lemma 4. *Given any function f of degree d, let f' be a map from $\mathbb{V}^d \to \mathbb{U}$ such that $f'(Z_1, ..., Z_d) = \sum_{\mathcal{S}\subseteq[d]} (-1)^{|\mathcal{S}|} f(\sum_{j\in\mathcal{S}} Z_j)$ for any $\{Z_j\}_{j\in[d]} \in \mathbb{V}^d$. Then f' is multilinear with respect to the d inputs. Moreover, if the characteristic of the base field $\mathbb{F}$ is 0 or greater than d, then f' is non-zero.*

Assuming the correctness of Lemma 4, it suffices to prove that f' enables computation designs that tolerates at least the same number of stragglers, and provides at least the same level of data privacy, compared to that of f. We prove this fact by constructing such computing schemes for f' given any design for f.

Note that f' is defined as a linear combination of functions $f(\sum_{j\in\mathcal{S}} Z_j)$, each of which is a composition of a linear map and f. Given the linearity of the encoding design, any computation scheme of f can be directly applied to any of these functions, achieving the same resiliency and privacy requirements. Since the decoding functions are linear, the same scheme also applies to linear combinations of them, which includes f'. Hence, the resiliency-privacy tradeoff achievable for f can also be achieved by f'. This concludes the proof.

H Proof of Lemma 4

We first prove that f' is multilinear with respect to the d inputs. Recall that by definition, f is a linear combination of monomials, and f' is constructed based on f through a linear operation. By exploiting the commutativity of these these two linear relations, we only need to show individually that each monomial in f is transformed into a multilinear function.

More specifically, let f be the sum of monomials $h_k \triangleq U_k \cdot \prod_{\ell=1}^{d_k} h_{k,\ell}(\cdot)$ where k belongs to a finite set, $U_k \in \mathbb{U}$, $d_k \in \{0, 1, ..., d\}$, and each $h_{k,\ell}$ is a linear map from $\mathbb{V}$ to $\mathbb{F}$. Let h'_k denotes the contribution of h_k in f', then for any $Z = (Z_1, ..., Z_d) \in \mathbb{V}^d$ we have

$$\begin{aligned} h'_k(Z) &= \sum_{\mathcal{S}\subseteq[d]} (-1)^{|\mathcal{S}|} h_k\left(\sum_{j\in\mathcal{S}} Z_j\right) \\ &= \sum_{\mathcal{S}\subseteq[d]} (-1)^{|\mathcal{S}|} U_k \cdot \prod_{\ell=1}^{d_k} h_{k,\ell}\left(\sum_{j\in\mathcal{S}} Z_j\right). \end{aligned} \tag{12}$$

By utilizing the linearity of each $h_{k,\ell}$, we can write h'_k as

$$\begin{aligned} h'_k(Z) &= U_k \cdot \sum_{\mathcal{S}\subseteq[d]} (-1)^{|\mathcal{S}|} \prod_{\ell=1}^{d_k} \sum_{j\in\mathcal{S}} h_{k,\ell}\left(Z_j\right) \\ &= U_k \cdot \sum_{\mathcal{S}\subseteq[d]} (-1)^{|\mathcal{S}|} \prod_{\ell=1}^{d_k} \sum_{j=1}^{d} \mathbb{1}(j\in\mathcal{S}) \cdot h_{k,\ell}\left(Z_j\right) \end{aligned} \tag{13}$$

Then by viewing each subset $\mathcal{S}$ of $[d]$ as a map from $[d]$ to $\{0,1\}$, we have[22]

$$\begin{aligned} h'_k(Z) = U_k \sum_{\boldsymbol{s}\in\{0,1\}^d} &\left(\prod_{m=1}^{d} (-1)^{s_m}\right) \\ &\cdot \prod_{\ell=1}^{d_k} \sum_{j=1}^{d} s_j \cdot h_{k,\ell}\left(Z_j\right) \\ = U_k \sum_{\boldsymbol{j}\in[d]^{d_k}} \sum_{\boldsymbol{s}\in\{0,1\}^d} &\left(\prod_{m=1}^{d} (-1)^{s_m}\right) \\ &\cdot \prod_{\ell=1}^{d_k} (s_{j_\ell} \cdot h_{k,\ell}\left(Z_{j_\ell}\right)). \end{aligned} \tag{14}$$

Note that the product $\prod_{\ell=1}^{d_k} s_{j_\ell}$ can be alternatively written as $\prod_{m=1}^{d} s_m^{\#(m \text{ in } \boldsymbol{j})}$, where $\#(m \text{ in } \boldsymbol{j})$ denotes the number of elements in $\boldsymbol{j}$ that equals m. Hence

$$\begin{aligned} h'_k(Z) = U_k \cdot \sum_{\boldsymbol{j}\in[d]^{d_k}} \sum_{\boldsymbol{s}\in\{0,1\}^d} &\left(\prod_{m=1}^{d} \left((-1)^{s_m} s_m^{\#(m \text{ in } \boldsymbol{j})}\right)\right) \\ &\cdot \prod_{\ell=1}^{d_k} h_{k,\ell}(Z_{j_\ell}) \\ = U_k \cdot \sum_{\boldsymbol{j}\in[d]^{d_k}} &\left(\prod_{m=1}^{d} \sum_{s\in\{0,1\}} (-1)^s s^{\#(m \text{ in } \boldsymbol{j})}\right) \\ &\cdot \prod_{\ell=1}^{d_k} h_{k,\ell}(Z_{j_\ell}). \end{aligned} \tag{15}$$

The sum $\sum_{s\in\{0,1\}} (-1)^s s^{\#(m \text{ in } \boldsymbol{j})}$ is non-zero only if m appears in $\boldsymbol{j}$. Consequently, among all terms that appear in (15), only the ones with degree $d_k = d$ and distinct elements in $\boldsymbol{j}$ have non-zero contribution. More specifically, [23]

$$h'_k(Z) = (-1)^d \cdot \mathbb{1}(d_k = d) \cdot U_k \cdot \sum_{g\in S_d} \prod_{j=1}^{d} h_{k,g(j)}\left(Z_j\right). \tag{16}$$

Recall that f' is a linear combination of h'_k's. Consequently, it is a multilinear function.

Now we prove that f' is non-zero. From equation (16), we can show that when all the elements Z_j's are identical, $f'(Z)$ equals the evaluation of the highest degree terms of f multiplied by a constant $(-1)^d d!$ with Z_j as the input for any j. Given that the highest degree terms can not be zero, and $(-1)^d d!$ is non-zero as long as the characteristic of the field $\mathbb{F}$ is greater than d, we proved that f' is non-zero.

[22] Here we define $0^0 = 1$.

[23] Here S_d denotes the symmetric group of degree d.

I Optimality in randomness

In this appendix, we prove the optimality of LCC in terms of the amount of randomness needed in data encoding, which is formally stated in the following theorem.

Theorem 3. *(Optimal randomness) Any linear encoding scheme that universally achieves a same tradeoff point specified in Theorem 1 for all linear functions f (i.e., (S, A, T) such that $K + T + S + 2A = N$) must use an amount of randomness no less than that of LCC.*

Proof. The proof is taken almost verbatim from [Huang, 2017], Chapter 3. In what follows, an $(n, k, r, z)_{\mathbb{F}_q^t}$ *secure RAID scheme* is a storage scheme over $\mathbb{F}_q^t$ (where $\mathbb{F}_q$ is a field with q elements) in which k message symbols are coded into n storage servers, such that the k message symbols are reconstructible from any $n - r$ servers, and any z servers are information theoretically oblivious to the message symbols. Further, such a scheme is assumed to use v random entries as keys, and by [Huang, 2017], Proposition 3.1.1, must satisfy $n - r \geq k + z$.

Theorem 4. *[Huang, 2017], Theorem 3.2.1. A linear rate-optimal $(n, k, r, z)_{\mathbb{F}_q^t}$ secure RAID scheme uses at least zt keys over $\mathbb{F}_q$ (i.e., $v \geq z$).*

Clearly, in our scenario $\mathbb{V}$ can be seen as $\mathbb{F}_q^{\dim \mathbb{V}}$ for some q. Further, by setting $N = n$, $T = z$, and $t = \dim \mathbb{V}$, it follows from Theorem 4 that any encoding scheme which guarantees information theoretic privacy against sets of T colluding workers must use at least T random entries $\{Z_i\}_{i \in [T]}$. □

J Optimality of LCC for Linear Regression

In this section, we prove that the proposed LCC scheme achieves the minimum possible recovery threshold R^* to within a factor of 2, for the linear regression problem discussed in Section 6.

As the first step, we prove a lower bound on R^* for linear regression. More specifically, we show that for any coded computation scheme, the master always needs to wait for at least $\lceil \frac{n}{r} \rceil$ workers to be able to decode the final result, i.e., $R^* \geq \lceil \frac{n}{r} \rceil$. Before starting the proof, we first note that since here we consider a more general scenario where workers can compute *any* function on locally stored coded sub-matrices (not necessarily matrix-matrix multiplication), the converse result in Theorem 2 no longer holds.

To prove the lower bound, it is equivalent to show that, for any coded computation scheme and any subset $\mathcal{N}$ of workers, if the master can recover $\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w}$ given the results from workers in $\mathcal{N}$, then we must have $|\mathcal{N}| \geq \lceil \frac{n}{r} \rceil$. Suppose the condition in the above statement holds, then we can find encoding, computation, and decoding functions such that for any possible values of $\boldsymbol{X}$ and $\boldsymbol{w}$, the composition of these functions returns the correct output.

Note that within a GD iteration, each worker performs its local computation only based on its locally stored coded sub-matrices and the weight vector $\boldsymbol{w}$. Hence, if the master can decode the final output from the results of the workers in a subset $\mathcal{N}$, then the composition of the decoding function and the computation functions of these workers essentially computes $\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w}$, using only the coded sub-matrices stored at these workers and the vector $\boldsymbol{w}$. Hence, if any class of input values $\boldsymbol{X}$ gives the same coded sub-matrices for each worker in $\mathcal{N}$, then the product $\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w}$ must also be the same given any $\boldsymbol{w}$.

Now we consider the class of input matrices $\boldsymbol{X}$ such that all coded sub-matrices stored at workers in $\mathcal{N}$ equal the values of the corresponding coded sub-matrices when $\boldsymbol{X}$ is zero. Since $\boldsymbol{0}^\top \boldsymbol{0} \boldsymbol{w}$ is zero for any $\boldsymbol{w}$, $\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{w}$ must also be zero for all matrices $\boldsymbol{X}$ in this class and any $\boldsymbol{w}$. However, for real matrices $\boldsymbol{X} = \boldsymbol{0}$ is the only solution to that condition. Thus, zero matrix must be the only input matrix that belongs to this class.

Recall that all the encoding functions are assumed to be linear. We consider the collection of all encoding functions that are used by workers in $\mathcal{N}$, which is also a linear map. As we have just proved, the kernel of this linear map is $\{\boldsymbol{0}\}$. Hence, its rank must be at least the dimension of the input matrix, which is dm. On the other hand, its rank is upper bounded by the dimension of the output, where each encoding function from a worker contributes at most $\frac{rdm}{n}$. Consequently, the number of workers in $\mathcal{N}$ must be at least $\lceil \frac{n}{r} \rceil$ to provide sufficient rank to support the computation.

Having proved that $R^* \geq \lceil \frac{n}{r} \rceil$, the factor of two characterization of LCC directly follows since $R^* \leq R_{\text{LCC}} = 2\lceil \frac{n}{r} \rceil - 1 < 2\lceil \frac{n}{r} \rceil \leq 2R^*$.

Note that the converse bound proved above applies to the most general computation model, i.e., there are no assumptions made on the encoding functions or the functions that each worker computes. If additional requirements are taken into account, we can show that LCC achieves the exact optimum recovery threshold (e.g., see [Yu et al., 2018]).

K Complete Experimental Results

In this section, we present the complete experimental results using the LCC scheme proposed in the paper, the gradient coding (GC) scheme [Tandon et al., 2017] (the cyclic repetition scheme), the matrix-vector multiplication based (MVM) scheme [Lee et al., 2015], and the uncoded scheme for which there is no data redundancy across workers, measured from running linear regression on Amazon EC2 clusters.

In particular, experiments are performed for the following 3 scenarios.

- Scenario 1 & 2: # of input data point $m = 8000$, # of features $d = 7000$.
- Scenario 3: # of input data point $m = 160000$, # of features $d = 500$.

In scenarios 2 and 3, we artificially introduce stragglers by imposing a 0.5 seconds delay on each worker with probability 5% in each iteration.

We list the detailed breakdowns of the run-times in 3 experiment scenarios in Tables 2, 3, and 4 respectively. In particular, the computation (comp.) time is measured as the summation of the maximum local processing time among all non-straggling workers, over 100 iterations. The communication (comm.) time is computed as the difference between the total run-time and the computation time.

Table 2: Breakdowns of the run-times in scenario one.

schemes	# batches/ worker (r)	recovery threshold	comm. time	comp. time	total run-time
uncoded	1	40	24.125 s	0.237 s	24.362 s
GC	10	31	6.033 s	2.431 s	8.464 s
MVM Rd. 1	5	8	1.245 s	0.561 s	1.806 s
MVM Rd. 2	5	8	1.340 s	0.480 s	1.820 s
MVM total	10	-	2.585 s	1.041 s	3.626 s
LCC	10	7	1.719 s	1.868 s	3.587 s

Table 3: Breakdowns of the run-times in scenario two.

schemes	# batches/ worker (r)	recovery threshold	comm. time	comp. time	total run-time
uncoded	1	40	7.928 s	44.772 s	52.700 s
GC	10	31	14.42 s	2.401 s	16.821 s
MVM Rd. 1	5	8	2.254 s	0.475 s	2.729 s
MVM Rd. 2	5	8	2.292 s	0.586 s	2.878 s
MVM total	10	-	4.546 s	1.061 s	5.607 s
LCC	10	7	2.019 s	1.906 s	3.925 s

Table 4: Breakdowns of the run-times in scenario three.

schemes	# batches/ worker (r)	recovery threshold	comm. time	comp. time	total run-time
uncoded	1	40	0.229 s	41.765 s	41.994 s
GC	10	31	8.627 s	2.962 s	11.589 s
MVM Rd. 1	5	8	3.807 s	0.664 s	4.471 s
MVM Rd. 2	5	8	52.232 s	0.754 s	52.986 s
MVM total	10	-	56.039 s	1.418 s	57.457 s
LCC	10	7	1.962 s	2.597 s	4.541 s