Round-Optimal Statistical Verifiable Secret Sharing with Polynomial Complexity

Chen-Da Liu-Zhang[1]
chen-da.liuzhang@hslu.ch

Elisaweta Masserova[2]
elisawem@andrew.cmu.edu

[1]School of Computer Science and Information Technology
Lucerne University of Applied Sciences and Arts, Switzerland

[2]Computer Science Department
Carnegie Mellon University, US

Abstract

Verifiable secret sharing (Chor, Goldwasser, Micali, and Awerbuch; FOCS'85) is a central primitive in secure computation, and its round complexity has been studied extensively. Rabin and Ben-Or [STOC'89] established the feasibility of statistical security up to $t < n/2$ corruptions, and Patra, Choudhary, Rabin, and Rangan [Crypto'09] showed that three sharing rounds are necessary. The work of Applebaum, Kachlon and Patra [STOC'23] introduced the first *round-optimal* VSS with three rounds of sharing and one round for reconstruction, albeit with exponential communication in the number of parties.

We give the first round-optimal statistical VSS that has polynomial communication and computation for $t < n/2$. This resolves the efficiency question left open by Applebaum, Kachlon and Patra. The protocol is obtained from a new notion of weak secret sharing, in which the reconstruction-time freedom of a corrupt dealer is limited to choosing verification keys, which is of independent interest.

Contents

1 Introduction **1**

1.1 Our contributions . . . 2

2 Technical overview **3**

2.1 The new weak secret sharing notion . . . 3

2.2 From key-uniform WSS to polynomial communication VSS . . . 4

2.3 Constructing the key-uniform WSS . . . 9

2.4 Replacing exhaustive search by polynomial-time search . . . 12

2.5 Privacy of the composition . . . 13

3 Related work **14**

4 Preliminaries **15**

5 Information checking **15**

5.1 A packed realization . . . 16

6 WSS with explicit verification keys **18**

7 A key-uniform WSS protocol **19**

7.1 Overview . . . 19

7.2 Protocol . . . 20

7.3 Security . . . 24

7.4 Efficient key search . . . 26

8 A $(3,1)$-round VSS from key-uniform WSS **26**

8.1 Overview . . . 27

8.2 Protocol . . . 27

8.3 Security . . . 28

8.4 Parameters . . . 30

9 AI Methodology **31**

A Proofs for Section 7 **34**

A.1 Authentication . . . 34

A.2 The row argument . . . 35

A.3 Privacy . . . 36

A.4 The polynomial search . . . 37

B Proofs for Section 8 **39**

B.1 Privacy and hidden points in the composition . . . 39

1 Introduction

Verifiable secret sharing (VSS), introduced by Chor, Goldwasser, Micali and Awerbuch [CGMA85], lets a dealer distribute a secret among n parties so that no coalition of t parties learns anything about it, while any dealer, honest or not, is *committed* to a single value that the honest parties will later reconstruct. VSS is the basic building block of information-theoretic multiparty computation (MPC) [BGW88, CCD88, RB89]: a protocol first has every party verifiably share its input and then computes on the shares, so the cost of VSS, and in particular its round complexity, is inherited by essentially every information-theoretic MPC protocol.

We work in the model of Rabin and Ben-Or [RB89]: n parties connected by private authenticated channels and a broadcast channel, facing a computationally unbounded, static, *rushing* adversary that corrupts up to $t < n/2$ of them and is allowed a negligible probability of error. This is the best resiliency for which information-theoretic VSS and MPC with guaranteed output delivery are possible at all, and in this setting the error is unavoidable: perfect security requires $t < n/3$ [BGW88].

The round complexity of VSS. A VSS protocol has a *sharing phase*, after which the dealer is committed, and a *reconstruction phase*, after which the honest parties output the committed value; we write (r_s, r_r) for a protocol with r_s sharing rounds and r_r reconstruction rounds. For perfect security with $t < n/3$ the picture is complete: three sharing rounds are necessary and sufficient [GIKR01], with one reconstruction round and polynomial complexity [FGG+06]. For statistical security with $t < n/2$ the sharing lower bound is the same, three rounds [PCRR09], and the known $(3, \cdot)$-round protocols are inefficient: Kumaresan, Patra and Rangan [KPR10] gave a $(3, 2)$-round protocol with computation exponential in n together with an efficient $(4, 2)$-round protocol, and Applebaum, Kachlon and Patra [AKP23] gave the first round-optimal $(3, 1)$-round protocol, again with complexity exponential in n, and used it to settle the round complexity of statistical MPC with optimal resiliency. They state explicitly that obtaining polynomially efficient protocols is left for future research. Thus, before this work, every three-round statistical VSS with optimal resilience $t < n/2$ had communication or computation exponential in the number of parties, even when allowing more than one reconstruction round.

The source of the difficulty is the single reconstruction round against a rushing adversary. In the statistical setting, shares are authenticated by information checking [RB89, CDD+99], where each verifier holds a private key; when the dealer and a share holder are both corrupt, they know every key and can make honest verifiers disagree about an opened share. Classical protocols repair this with an extra reconstruction round in which the verifiers vote. The protocol of [AKP23] instead builds a new statistical signature whose single-round opening yields a common verdict, but its verifiers are committees of t parties, which is where the exponential cost comes from.

This work. We give the first statistical VSS with three sharing rounds, one reconstruction round, resiliency $t < n/2$, and communication and computation polynomial in n and the security parameter. Sharing privacy is perfect. The construction is organized around a new notion of weak secret sharing, which we call *key-uniform* weak secret sharing, and a conditional compiler to VSS that preserves round complexity, provided the key-uniform guarantees hold jointly and the composed sharing phase remains private; both are of independent interest and are stated in Section 1.1.

1.1 Our contributions

All results are in the model above: n parties, private channels and broadcast, a computationally unbounded static rushing adversary corrupting $t < n/2$ parties, and a statistical security parameter κ.

Theorem 1.1 (Main result, informal)**.** *There is a verifiable secret sharing protocol with three sharing rounds and one reconstruction round, tolerating $t < n/2$ corruptions, with perfect privacy of the sharing phase, correctness and commitment error $2^{-\kappa}$, and communication and computation polynomial in n, κ and the secret length.*

By the lower bound of [PCRR09] the number of sharing rounds is optimal, and one reconstruction round is trivially optimal, so the protocol is round-optimal. Theorem 1.1 resolves the efficiency question left open in [AKP23].

The protocol is obtained in two steps. The first is a definitional one. In a weak secret sharing scheme (WSS), the dealer is committed to a value ρ, but a corrupt dealer may still decide during reconstruction whether the honest parties output ρ or reject. We consider WSS schemes whose reconstruction has a specific form: every party broadcasts an opening together with a vector of *verification keys* fixed after the first sharing round, and each party's output is a public function of its saved sharing state, the transcript, the openings and the keys. For such schemes we define *key-uniform* WSS by two requirements: the weak commitment guarantee holds simultaneously for every assignment of corrupt keys, and for any fixed assignment the decision to reject cannot be influenced by the corrupt parties' openings. Neither property is implied by ordinary weak commitment, and together they say that the only reconstruction-time freedom a corrupt dealer retains is the choice of its keys.

Theorem 1.2 (Conditional compiler, informal)**.** *Let Π be a key-uniform WSS with three sharing rounds and one reconstruction round whose verification keys have length independent of the secret length. Running $n+1$ instances of Π as in Section 8, with one extra broadcast per party in the second sharing round, gives a protocol with three sharing rounds and one reconstruction round that is correct and committing with error ε whenever the $n+1$ instances satisfy the key-uniform guarantees jointly in the composed execution, except with probability ε. Its local computation is polynomial whenever Π admits a polynomial-time search over corrupt key assignments. If the composed sharing phase is also private, the protocol is a VSS.*

The construction removes the remaining freedom of a corrupt dealer by fixing the corrupt parties' keys during the sharing phase, without an additional round and without an exponential search; the technical overview explains how. The joint guarantee and the privacy of the composition do not follow from standalone key-uniformity, because each party's masked keys include its keys for the instance that shares the mask; we prove both for our concrete scheme.

Theorem 1.3 (Key-uniform WSS, informal)**.** *There is a key-uniform WSS with three sharing rounds and one reconstruction round, tolerating $t < n/2$ corruptions, with perfect sharing privacy, error $2^{-\kappa}$, polynomial communication and computation, and a polynomial-time search over corrupt key assignments.*

The scheme follows the row-and-pad structure of earlier honest-majority VSS [KPR10], with the shares authenticated by a variant of information checking in which verification keys may be published after the openings they judge have been fixed. Theorem 1.1 follows from Theorem 1.2 instantiated with the scheme of Theorem 1.3, together with the privacy and joint authentication arguments for that composition in Section 8.

2 Technical overview

The key towards achieving a $(3, 1)$-round VSS with polynomial communication lies in identifying the right notion of weak secret sharing (WSS), which serves as the main building block in our VSS. Informally, WSS satisfies the same properties as VSS, except that it leaves one choice to a dishonest dealer: The adversary may decide whether reconstruction returns the committed value or rejects during the reconstruction phase itself. Our starting point is to isolate *where* this choice is made.

We define a new notion, *key-uniform weak secret sharing* (kuWSS), which explicitly defines parties' verification keys. Once these keys are fixed for the honest parties, the decision to reject is already determined by sharing. Further, changing the keys of corrupted parties cannot change the value of a successful reconstruction.

Curiously, this abstraction gives a very simple way to obtain VSS with polynomial communication (albeit exponential computation): **(1)** share the secret via kuWSS, **(2)** share kuWSS's verification keys themselves (again, via kuWSS), and **(3)** reconstruct a canonical assignment of verification keys via an exhaustive local search. The second and third step effectively ensure that kuWSS's keys were fixed by the end of the sharing phase, and are thus safe to use when reconstructing the original secret. Note that the sharings of kuWSS's verification keys have verification keys of their own. Sharing these new keys directly would therefore appear to require yet another layer of kuWSS instances, and so on. In the following, we discuss how to avoid this recursion, as well as the remaining subtleties of our compiler and of our instantiation of the key-uniform WSS.

We consider n parties $P_1, \ldots, P_n$, secure pairwise channels, and ideal broadcast. A static, computationally unbounded, rushing adversary corrupts a set $\mathcal{C}$ of size at most $t < n/2$; $\mathcal{H} = [n] \setminus \mathcal{C}$ denotes the set of honest parties. We denote by $\mathbb{E}$ a finite field.

2.1 The new weak secret sharing notion

Separating values from rejection. An ordinary WSS allows a dealer to share a secret $x \in \mathbb{E}^h$, with privacy against any coalition of at most t parties when the dealer is honest. The honest parties' joint view at the end of sharing fixes a value $\rho \in \mathbb{E}^h$, with $\rho = x$ for an honest dealer. Except with small probability, each honest party reconstructs either ρ or $\perp$, and all honest parties reconstruct x if the dealer is honest. For a corrupt dealer, however, whether reconstruction returns ρ or $\perp$ may depend on the adversary's reconstruction messages. This is in contrast to VSS, which additionally requires a common reconstruction outcome, to be fixed by sharing.

Consider a WSS whose reconstruction consists of one broadcast round followed by local decoding. Each honest party P_j broadcasts its original verification vector $\kappa_j \in \mathbb{E}^\ell$ and its prescribed opening data ω_j. Each honest party P_i then applies a decoding function Out to its saved sharing state σ_i, the broadcast verification vectors, and the broadcast opening data.

To formulate our version of the WSS, dubbed *key-uniform WSS*, we consider evaluating the same decoder on an arbitrary assignment of verification vectors $K = (k_1, \ldots, k_n)$ and an arbitrary opening transcript $\tau = (w_1, \ldots, w_n)$:

$$\mathsf{Out}(\sigma_i, K; \tau) \in \mathbb{E}^h \cup \{\perp\}.$$

We call K *honest-consistent* if $k_j = \kappa_j$ for every $j \in \mathcal{H}$. Similarly, we call τ *honest-consistent* if $w_j = \omega_j$ for every $j \in \mathcal{H}$.

Key-uniform WSS has two parts. There is a value ρ fixed by the honest parties' joint state at the end of sharing such that, for every adversarial reconstruction strategy, with probability at least $1 - \varepsilon$,

we require that the following properties hold *simultaneously for every honest-consistent assignment* K:

(a) **Uniform weak commitment.** On the actual opening transcript τ, all honest parties obtain the same output in $\{\rho, \perp\}$. If the dealer is honest, that output is its input.

(b) **Sharing-determined rejection.** For each honest observer P_i and any two honest-consistent opening transcripts τ_1, τ_2,

$$\mathsf{Out}(\sigma_i, K; \tau_1) = \perp \quad \Longleftrightarrow \quad \mathsf{Out}(\sigma_i, K; \tau_2) = \perp.$$

Finally, we require that each party P_i knows its verification key κ_i by the end of the first round.

Beyond ordinary WSS. Ordinary WSS guarantees weak commitment for the reconstruction actually carried out by the parties. Here, we additionally require that the same opening transcript can be decoded under every honest-consistent assignment of verification vectors: except with probability ε, all such assignments yield either the same sharing-fixed value ρ or $\perp$, with agreement among honest observers for each assignment. In particular, the probability that this guarantee fails for even one assignment is at most ε.

Property (b) imposes a further requirement that ordinary weak commitment does not provide. For each fixed honest-consistent assignment K, whether decoding rejects must be independent of the corrupt parties' opening messages. Thus, **once K is fixed, the adversary cannot change which assignments are rejected during reconstruction**.

2.2 From key-uniform WSS to polynomial communication VSS

The above suggests a way to close the gap between the key-uniform WSS and VSS: we must fix K during the sharing phase.

One mask per party. We can do so by sharing the verification keys themselves using key-uniform WSS. This seems to create a recursive problem: the additional WSS instances have verification keys of their own, which must also be fixed. Our construction resolves this apparent recursion using only one additional WSS instance per party.

Instead of sharing each verification vector separately, each party samples a random mask of the required length and shares it using a single key-uniform WSS instance. It then collects its verification vectors across *all instances* into one vector and publishes a masked version of this vector during the sharing phase. Once the shared mask value is fixed, the published masked vector uniquely determines the corresponding verification vectors.

Crucially, this vector includes the party's verification keys for the very *kuWSS instance used to share the mask itself*. This is possible if the length of a kuWSS verification vector is independent of the length of its shared input: making the mask long enough to cover all these keys does not create longer keys that would require a still larger mask. We thus need only $n + 1$ such kuWSS instances: one for the original secret, denoted kuWSS_0 and one for each party's mask. We denote the kuWSS instance used for the sharing of the secret as kuWSS_0, and the kuWSS instance used for sharing of P_i's mask as kuWSS_i.

We briefly note that the masking is not a straightforward one-time-pad argument: the vector P_i masks includes its keys for the very instance kuWSS_i that shares the mask, so the mask is correlated

with part of what it hides, and key-uniformity alone does not rule out leakage. Privacy is therefore proved for the composition with our concrete WSS (Section 2.5); correctness and commitment of the compiler, discussed next, use only the key-uniform properties.

A natural reconstruction strategy is to recover the shared masks, use them to recover the verification vectors from the published masked vectors, and then reconstruct the original secret. The difficulty is that kuWSS allows mask reconstruction to fail, potentially leaving some verification vectors unavailable. The recovery procedure described next uses the mask-sharing instances to construct an **honest-consistent verification-key assignment determined by the sharing phase**, without requiring every mask to be recovered. Applying the key-uniform guarantee to the original instance under this assignment then gives a reconstruction output fixed by sharing, including its decision to output $\perp$.

The first recovery attempt. To get started, we only need an assignment that preserves the honest parties' verification vectors; the corrupt parties' coordinates may be arbitrary. We obtain such an assignment by asking every party to simply announce its mask during reconstruction, together with its opening data for all kuWSS instances $\mathsf{kuWSS}_0, \ldots, \mathsf{kuWSS}_n$ (see Protocol 1 for this checkpoint). Corrupt parties may announce false masks or remain silent. Nevertheless, the honest announcements recover the honest parties' original verification vectors and therefore give us an honest-consistent starting assignment.

Protocol 1: VSS from key-uniform WSS

Parameters and inputs. Let $m = (n+1)\ell$, where ℓ denotes the length of a party's verification vector in a single kuWSS instance. Let $\lambda \in \mathbb{E} \setminus \{0\}$ denote a publicly known value.

Round 1. Dealer D initiates a key-uniform WSS for x, denoted by kuWSS_0. Each P_i samples $R_i \leftarrow \mathbb{E}^m$, and initiates a key-uniform WSS for R_i, denoted by kuWSS_i.

Round 2. Let κ_i^j denote the P_i's verification vector output in kuWSS_j, for $j \in \{0, \ldots, n\}$. Let

$$X_i = (\kappa_i^0, \ldots, \kappa_i^n) \in \mathbb{E}^m.$$

Each P_i broadcasts $C_i = X_i + \lambda R_i$.

Reconstruction broadcast. Each P_i broadcasts its mask $r_i = R_i$ together with its prescribed openings in all kuWSS instances. Save the actual kuWSS opening transcripts $\tau_0, \ldots, \tau_n$. All further work is local (described separately).

For a party P_i that broadcast the masked vector C_i in the second round of the sharing phase and the claimed mask r_i during the reconstruction phase, define its raw key assignment as $k_i^{\text{raw}} = C_i - \lambda r_i$. If either C_i or r_i is missing or malformed, let $k_i^{\text{raw}} = \mathsf{bad}$. This assignment preserves every honest key.

Each party P_i uses this same raw assignment to attempt reconstruction of every mask-sharing kuWSS instance. For instance kuWSS_j, this is done via $\mathsf{Out}(\sigma_i^j, K^j, \tau_j)$, where σ_i^j is P_i's state in kuWSS_j, K^j consists of the block corresponding to the instance kuWSS_j from each raw verification key, and τ_j is the opening transcript of kuWSS_j.

Because the raw assignment is honest-consistent, every honestly shared mask is recovered correctly. An instance shared by a corrupt party may reject, but any value it returns equals its sharing-fixed value ρ_j. This value need not equal the corrupt party's announced mask r_j. We cache every non-$\perp$

result ρ_j and compute its corresponding verification key

$$k_j = C_j - \lambda\rho_j.$$

After this procedure, let U contain the indices of masks not yet cached. Since every honestly shared mask has been recovered, U *contains only corrupt parties' indices.*

Each cached value is fixed by sharing, but the set of masks recovered in this first attempt may still depend on the corrupt parties' announced masks, through the resulting raw key assignment. To obtain a final key assignment fixed by sharing despite this dependence, in the next step we try alternative candidate keys for parties in U, using the cached keys elsewhere and rerunning the local decoders with the same received opening data (i.e., τ_i for kuWSS_i). Since U contains only corrupt parties' indices, varying these coordinates preserves honest-consistency. A different assignment may allow additional masks to be recovered, but it may also cause a previously accepting instance to reject[1]. We therefore examine how reconstruction behaves as the unresolved key coordinates vary.

Closing the rejection dependencies. Let Z record the key coordinates currently allowed to vary, initially $Z = U$. Outside Z, use the keys k_j derived from the cached masks via $k = C - \lambda\rho$. We check whether each mask instance kuWSS_i with $i \notin Z$ continues to accept under every allowed choice of the keys in Z.

If such an instance rejects under some choice, the key-uniform guarantees imply that its dealer P_i is corrupt. We may therefore also allow P_i's candidate key to vary while preserving honest-consistency. Add i to Z and repeat the check: allowing another coordinate to vary introduces more possible assignments and may cause further instances to reject.

Note that adding i to Z does not erase its cached mask or put it back into U. The cache records recovered mask values, whereas Z records key coordinates allowed to vary. Once no further index is added to Z, every mask instance indexed outside Z accepts under every allowed completion.

We then keep Z fixed and search for accepting completions for the still-missing masks. Every success supplies a sharing-fixed mask value. If the cache grows, we restart the closure from the smaller missing set U, retaining all recovered values.

In more detail: A well-formed masked vector C_j broadcast during the sharing phase permits a corresponding key coordinate in $\mathcal{D}_j = \mathbb{E}^m \cup \{\mathsf{bad}\}$, while a missing or malformed C_j forces $\mathcal{D}_j = \{\mathsf{bad}\}$. We write

$$K^{\mathrm{raw}} = (k_1^{\mathrm{raw}}, \ldots, k_n^{\mathrm{raw}}).$$

Whenever a mask value ρ_j of party P_j is cached, its corresponding key is

$$k_j = \begin{cases} C_j - \lambda\rho_j, & C_j \in \mathbb{E}^m, \\ \mathsf{bad}, & \text{otherwise.} \end{cases}$$

Let $k = (k_j)_{j\in[n]\setminus U}$ denote the partial key assignment derived from the cached masks. For $Z \supseteq U$, we form candidate assignments by retaining these cached keys outside Z and allowing the coordinates in Z to vary within their domains. Thus,

$$\mathcal{K}(Z,k) = \left\{ K' = (k'_1, \ldots, k'_n) : \begin{array}{ll} k'_j = k_j, & j \notin Z, \\ k'_j \in \mathcal{D}_j, & j \in Z \end{array} \right\}. \tag{1}$$

[1]By the guarantees of the key-uniform WSS, this can happen only for the instances dealt by corrupt parties.

Note that (1) every required k_j is available because $U \subseteq Z$, and (2) even if a coordinate in Z has a cached key, that coordinate remains free in these candidate assignments.

For an assignment K, let K^i contain the block corresponding to kuWSS_i from each coordinate of K[2]. For $a \in \{0, \ldots, n\}$, write

$$D_{i,a}(K) = \mathsf{Out}(\sigma_i^a, K^a; \tau_a).$$

Here, τ_a is the actual opening transcript received for kuWSS_a during the reconstruction round. We describe the complete local reconstruction procedure in Protocol 2.

Protocol 2: Canonical local recovery for party P_i

All scans and exhaustive searches use fixed public orders.

1. **Initialize the cache.** Evaluate $D_{i,a}(K^{\mathrm{raw}})$ for every $a \in [n]$. Cache each non-$\perp$ mask output ρ_a and compute its corresponding key k_a as above. Let U contain the remaining indices.
2. **Close.** Set $Z = U$. For each $a \in [n] \setminus Z$, check if some $K' \in \mathcal{K}(Z, k)$ satisfies $D_{i,a}(K') = \perp$. If so, add a to Z and restart the scan. Stop when a complete scan makes no addition.
3. **Recover.** Keep Z fixed and record $U_{\mathrm{old}} = U$. For each $a \in U_{\mathrm{old}}$, search for $K' \in \mathcal{K}(Z, k)$ with $D_{i,a}(K') \neq \perp$. If one exists, cache its output ρ_a, compute k_a, extend k by this entry, and remove a from U.
4. **Restart or finish.** If $U \neq U_{\mathrm{old}}$, return to step 2. Otherwise define $K^\star = (k_1^\star, \ldots, k_n^\star)$ by

$$k_j^\star = \begin{cases} \mathsf{bad}, & j \in Z, \\ k_j, & j \notin Z, \end{cases}$$

and output $D_{i,0}(K^\star)$.

Note that each restart in Protocol 2 adds at least one previously missing mask to the cache, so there are at most n restarts.

Why every search is safe. Assume that properties (a) and (b) of key-uniform WSS hold simultaneously for the $n+1$ instances in the actual composed execution. Initialization recovers every honest party's mask, and every successful recovery returns its sharing-fixed value. Thus $U \subseteq \mathcal{C}$.

Inductively, suppose $Z \subseteq \mathcal{C}$. Every honest coordinate lies outside Z and is fixed to the key derived from its owner's correctly recovered mask. Every completion is therefore honest-consistent. By the security of kuWSS, an honestly dealt mask instance cannot reject under such an assignment. Closure consequently adds only corrupt indices, and successful searches recover only sharing-fixed values. Hence, during every closure and recovery pass,

$$U \subseteq Z \subseteq \mathcal{C}.$$

[2]A coordinate of K corresponds to a single party's verification vector, which in turn contains this party's verification keys for different kuWSS instances.

Why the final assignment is fixed by sharing. The preceding argument shows that every recovered mask value is sound, hence every recovered key assignment is honest-consistent. It remains to show that the final key assignment K^* from Protocol 2 does not depend on which masks were recovered in the first attempt, and hence on the corrupt parties' reconstruction announcements.

Fix the honest parties' joint sharing view S. For each $a \in \{0, \ldots, n\}$, let $\rho_a(S)$ denote the sharing-fixed value guaranteed by key-uniform WSS for kuWSS_a: under its guarantees, every successful decoding with an honest-consistent key assignment returns $\rho_a(S)$. For $a \in [n]$, this is the committed mask value, and $\rho_a(S) = R_a$ when P_a is honest. For $a = 0$, it is the committed value of the original instance.

These values are defined for analysis purposes; the algorithm above need not have recovered them.

Similarly for analysis purposes, we define the complete collection of keys

$$k_j(S) = \begin{cases} C_j - \lambda\rho_j(S), & C_j \in \mathbb{E}^m, \\ \mathsf{bad}, & \text{otherwise}, \end{cases} \qquad j \in [n],$$

and write

$$K(S) = (k_1(S), \ldots, k_n(S)). \tag{2}$$

This assignment is determined by sharing, even though the algorithm may not recover every mask needed to compute it. Note that every cached key agrees with its corresponding $k_j(S)$. Further, for honest j, we have $\rho_j(S) = R_j$ and hence $k_j(S) = X_j$.

For any honest-consistent candidate key assignment K, property (b) makes rejection of kuWSS_a under the projected assignment K^a independent of the corrupt parties' opening messages. Since the honest openings are determined by S, the rejection decision depends only on S and K. Denote this decision by $b_a(S, K)$, where 1 means rejection.

Property (a) makes this bit common to honest parties and gives

$$D_{i,a}(K) = \begin{cases} \perp, & b_a(S, K) = 1, \\ \rho_a(S), & b_a(S, K) = 0, \end{cases}$$

for every honest-consistent K, using the actual received opening transcript.

Let $\mathcal{K}_S(Z)$ denote the completion domain obtained using the full sharing-fixed keys:

$$\mathcal{K}_S(Z) = \left\{ K' = (k'_1, \ldots, k'_n) \in \prod_{j=1}^{n} \mathcal{D}_j : k'_j = k_j(S) \text{ for every } j \notin Z \right\}.$$

For $Z \subseteq \mathcal{C}$, define

$$\Phi_S(Z) = Z \cup \left\{ a \in [n] : \exists K' \in \mathcal{K}_S(Z),\ b_a(S, K') = 1 \right\}. \tag{3}$$

For $Z \subseteq \mathcal{C}$, every assignment in $\mathcal{K}_S(Z)$ is honest-consistent: every honest coordinate lies outside Z and is fixed to its original key $k_j(S) = X_j$. An honestly dealt mask instance therefore cannot reject under any such assignment. Thus, every index added by Φ_S is corrupt, and $\Phi_S(Z) \subseteq \mathcal{C}$. Moreover, it holds that

$$\mathcal{K}(Z, k) = \mathcal{K}_S(Z).$$

Where $K(Z, k)$ is defined in Equation 1. The operator Φ_S therefore formalizes the rule used in the closure step of Protocol 2: an index is added to Z if its mask instance can reject under some allowed

completion. The closure step repeatedly applies this rule, starting from the current missing set U, until no further indices can be added.

By definition, $\Phi_S(Z)$ contains Z, so applying Φ_S can only add indices. Moreover, enlarging Z enlarges the set of completions, so Φ_S is monotone. Let Z_0 be its least fixed point starting from the empty set. Note that both the operator Φ_S and Z_0 are determined by the sharing phase.

By monotonicity, each execution of the closure step computes the least fixed point of Φ_S containing the missing set U at the start of that execution. At termination, every still-missing mask instance rejects under every assignment in $\mathcal{K}_S(Z)$. This domain includes $K(S)$ (defined in Equation 2): it agrees with the required keys outside Z and assigns allowed values to the free coordinates inside Z. Hence every $a \in U$ rejects under $K(S)$. Since $\mathcal{K}_S(\emptyset) = \{K(S)\}$, this implies $U \subseteq \Phi_S(\emptyset)$.

Therefore

$$U \subseteq \Phi_S(\emptyset) \subseteq Z_0.$$

The terminal closure is consequently contained in Z_0. Conversely, it is a fixed point and therefore contains the least fixed point Z_0. Hence the terminal set is exactly Z_0, independently of the initial announcements.

The final assignment is therefore

$$k_j^\star = \begin{cases} \mathsf{bad}, & j \in Z_0, \\ k_j(S), & j \notin Z_0. \end{cases}$$

It is sharing-fixed and honest-consistent. Applying the key-uniform guarantee to kuWSS_0 fixes its rejection decision to $b_0(S, K^\star)$ and, if it accepts, its output to $\rho_0(S)$. Thus the entire reconstruction output is fixed by sharing.

The construction of Protocols 1 and 2 has polynomial communication whenever the underlying key-uniform WSS does, but the exhaustive searches make the reconstruction exponential. There are three points left to discuss: how to obtain a key-uniform WSS (Section 2.3), how that WSS supports polynomial-time search (Section 2.4), and why the masked keys stay private in the composition (Section 2.5).

2.3 Constructing the key-uniform WSS

In this section we describe how to obtain the key-uniform WSS at a high level. See Section 7 for details about the construction.

Warm-up: WSS from digital signatures. Suppose first that the parties have ordinary digital signatures, with public verification. The dealer D samples a symmetric bivariate polynomial B of degree t in each variable with $B(0,0) = s$, and each party P_i (including the dealer), a *row holder*, receives the row $g_i(Y) = B(i, Y)$. Any t rows reveal nothing about s, any $t+1$ rows determine B, and two rows overlap in one value, $B(i,j) = B(j,i)$. At least $n - t \geq t + 1$ row holders are honest, so the honest rows determine B as soon as they are pairwise consistent at these overlaps, and the protocol checks this without revealing the overlaps. The warm-up ignores missing or malformed messages, which the protocol of Section 7 handles explicitly.

Round 1. D sends P_i its row g_i, with a signature on each value $g_i(j)$. Each P_i samples a random pad r_{ij} for every j and sends it to P_j, signed.

Round 2. P_i broadcasts $a_{ij} = g_i(j) + r_{ij}$ and $b_{ij} = g_i(j) + r_{ji}$, using its own pad and the pad received from P_j. If P_i and P_j are consistent then $a_{ij} = b_{ji}$, which constitutes a public witness of $g_i(j) = g_j(i)$ that reveals neither value; a mismatch is a *dispute*.

Round 3. For a disputed pair, P_i and P_j open their overlaps $g_i(j)$ and $g_j(i)$ together with the dealer's signatures on them. A party whose row is not a valid degree-t row broadcasts the received values with the dealer's signatures on them.

Reconstruction. Each P_i broadcasts its row f_i and, for every j, the pad r_{ji} it received from P_j with P_j's signature. A party first checks the round-three openings: if two overlaps carrying valid dealer signatures differ, or a row carrying valid dealer signatures is not of degree t, the dealer is rejected. Otherwise it keeps row f_i if f_i has degree t, agrees with every opened overlap, and is consistent with the pads, that is, $f_i(j) + r_{ij} = a_{ij}$ for every j that opened the pad r_{ij} with a valid signature of P_i. It then interpolates any $t + 1$ kept rows at 0.

With an honest dealer no two honest parties dispute, so nothing about honest rows is revealed; a corrupt party may dispute an honest one, but the overlap then opened is one it already knows. Further, if the sharing succeeds, every honest row is kept even with a corrupt dealer, and the pad checks and opened overlaps force any kept row to agree with the honest rows at $t + 1$ overlaps, so it lies on the polynomial the honest rows define; all parties therefore reconstruct the same value.

The signatures achieve three points: The dealer's signatures make the dealer accountable: an opened overlap or an invalid row with a valid signature is proof against the dealer. The parties' signatures on pads make the row holders accountable: a party can only open the pad it actually received, so it cannot tailor a pad to make a bad row pass. And in both cases every party verifies the same signature, so all honest parties reach the same verdict.

Information checking. Information checking (IC) is the information-theoretic stand-in for a signature: the sender gives the holder a message and gives every verifier P_w a private key, and at opening each verifier checks the holder's candidate against its own key. Our IC instance runs in three phases: one round of *distribution*, in which the sender gives the holder the message and the verifiers their keys; two rounds of *verification*, in which the holder and the sender exchange broadcasts that commit the sender to the message; and one round of *reveal*, in which the holder broadcasts its candidate. The reveal may run in parallel with the second verification round (Section 5).

IC guarantees *nonrepudiation*, i.e., an honest holder's candidate is accepted by every honest verifier, and *unforgeability*, i.e., a wrong candidate for an honest sender's message is rejected by every honest verifier. What IC lacks is a common verdict: when sender and holder are both corrupt they know all keys and can make honest verifiers disagree. In the following, we use the standard notion of IC (see Section 5) with one presentational change and one observation.

(a) The verifier's check is written as an explicit public algorithm $\mathsf{Open}(\text{candidate}, \text{key}_w)$, so that anyone who learns key_w can recompute P_w's verdict. Note that any IC verifier decides by applying a deterministic rule to its key; the standard definition typically leaves this rule implicit.

(b) Unforgeability is unaffected if key_w is published *after* the candidate it judges has been fixed: the acceptance event is determined at that moment, and nothing learned later changes its probability. So a key can be released as soon as there is nothing left for it to be forged against.

The WSS. Replacing each signature by an IC instance, called a *label*, with the same sender, holder and message, costs one change to the protocol and one decision per label. The distribution and the two verification rounds of all labels run in the three sharing rounds, and since the reveal may run in parallel with the second verification round, a label can be opened in sharing round three.

The change: the dealer repairs rows. In the warm-up a party with an invalid row proves it by publishing the row with its signatures. In our WSS rows are opened only at reconstruction, with private keys (see LATE labels below), so an invalid row cannot be exhibited in round three; and if the dealer is corrupt every honest row must nevertheless be usable at reconstruction, since commitment rests on the honest rows.

To address this, the dealer takes part in the comparison: in round two it also broadcasts its own versions a_{ij}^D, b_{ij}^D of the masked overlaps, which requires it to hold a copy of every pad, sent to it in round one. A party that flags its row as invalid in round two, or whose broadcast disagrees with the dealer's, has its row *repaired*: in round three the dealer broadcasts a set U of such rows and the rows u_i, $i \in U$, which replace the received ones. (If an honest dealer repairs the row of a corrupt party that falsely flagged it, nothing is lost.)

A repaired row u_j must be checked against the unrepaired rows i, and this is where the pad copies are used: the dealer opens its copy of r_{ij} in round three, and the check is $u_j(i) + r_{ij} = a_{ij}$. The opened copy can be trusted for the rows that matter. It is authenticated with P_i as sender, so if P_i is honest the dealer can open only the genuine pad, or fail and be rejected; and since row i was not repaired, a_{ij} agrees with the dealer's own a_{ij}^D, so the genuine pad ties $u_j(i)$ to P_i's overlap $g_i(j)$. The labels are thus: overlaps $g_i(j)$ (labels S_{ij}, sender D, holder P_i) and the dealer's pad copies (labels Q_{ij}, sender P_i, holder D), opened in round three; rows (labels $W_{i,\ell}$, one per coefficient, sender D, holder P_i) and the pads held by neighbours (labels R_{ij}, sender P_i, holder P_j), opened at reconstruction. The decision per label is when its keys may be released.

EARLY labels (S, Q). Their candidates are broadcast in sharing round three, so the values are fixed before reconstruction. Whether a label must be revealed is a public function of the round-two broadcasts (for S_{ij}, a dispute between P_i and P_j; for Q_{ij}, that row j was repaired and row i was not), and a required label for which no corresponding candidate was published in round three counts as a failed opening. From then on nothing can be added: at reconstruction the parties publish keys, but the only candidate ever judged for an EARLY label is the one broadcast in round three. Each party P_w publishes its keys for these labels, concatenated into its *verification vector* κ_w, and an opening is accepted iff at least $t + 1$ of the n published keys accept it, where each vote is computed by the public Open. The threshold works because at most t of the n published keys are corrupt. An honest holder's genuine opening is accepted by every honest key, hence by at least $n - t \geq t + 1$ keys, whatever the corrupt keys say; a wrong opening against an honest sender is rejected by every honest key, which by (b) remains true after the keys are published, so at most t keys accept it. In both cases the honest keys alone decide, and since all parties count the same published keys, they reach the same verdict.

LATE labels (W, R). Their candidates are broadcast at reconstruction. Thus, their keys must stay private, and every party must judge them with its own key, so verdicts may differ between parties. However, no common verdict is needed: These openings only decide whether a party keeps a given row, and we saw in the warm-up that any kept row lies on the polynomial defined by the honest rows, whichever corrupt rows a party keeps; repaired rows are public and always kept. Note that honest rows are always kept: an honest party's own openings are accepted by nonrepudiation, and a corrupt neighbour cannot open a wrong pad against it by unforgeability.

The result is a WSS whose reconstruction messages are openings plus the published keys κ_w. The

division of labels gives the properties of key-uniform WSS that were outlined in Section 2.1. First, the decision to reject depends only on the sharing transcript and the keys published at reconstruction: it counts EARLY votes, each computed by the public Open from a published key and a candidate that was frozen during sharing, so neither the reconstruction openings nor any party's private state can affect it. Second, the value returned when not rejecting does not depend on the corrupt keys at all: as long as the honest parties publish their own keys, those keys alone force every kept row onto the honest polynomial. Together, these say that the corrupt keys can only decide *whether* the committed value is returned, never *which* value. And that decision is indeed still open to the adversary: a corrupt dealer colluding with a corrupt party can choose the keys that party publishes, and a suitable choice moves the vote on some EARLY opening across the threshold $t+1$, which turns a rejection into an acceptance or vice versa.

2.4 Replacing exhaustive search by polynomial-time search

Efficient evaluation is not efficient search. The recovery procedure of Protocol 2 interacts with the WSS through two questions only. Given the free set Z and the keys cached outside it, does some completion of the free coordinates make instance kuWSS_a reject, and does some completion make it accept? Evaluating Out on one assignment is cheap, but there are exponentially many completions, and Definition 6.2 gives no reason to believe that the search can be shortcut. We therefore add a third, purely computational property to key-uniform WSS.

(c) **Efficient key search.** There is a deterministic polynomial-time algorithm Search which, on input a party's state σ_i, the openings τ, a set $Z \subseteq [n]$, keys $c_j \in \mathbb{E}^\ell \cup \{\mathsf{bad}\}$ for $j \notin Z$, and a bit $b \in \{\mathsf{rej}, \mathsf{acc}\}$, returns an assignment K that agrees with the c_j outside Z and satisfies $\mathsf{Out}(\sigma_i, K; \tau) = \perp$ if $b = \mathsf{rej}$ and $\mathsf{Out}(\sigma_i, K; \tau) \neq \perp$ if $b = \mathsf{acc}$, or returns none if no such assignment exists.

The answer has to be exact on every input, adversarial or malformed, and Search does not know which parties are honest. Note that our exhaustive enumeration is a valid, albeit exponential-time implementation, so (c) asks for efficiency and nothing else.

The same reconstruction, now efficient. With (c) in hand, we simply replace the two exhaustive searches in Protocol 2 by calls to Search: the closure step calls $\mathsf{Search}(\cdot, \mathsf{rej})$ for each $a \notin Z$, and a recovery pass calls $\mathsf{Search}(\cdot, \mathsf{acc})$ for each missing mask and, on a witness, runs Out once to obtain the mask. Note the following two details:

First, the recovery manipulates packed assignments, with one block per instance, whereas Search runs on a single instance; so a query is projected onto the blocks of kuWSS_a, fixing a coordinate to bad whenever the corresponding deposit C_j is malformed, and a witness is lifted back by keeping the fixed coordinates, turning a free coordinate returned as bad into a packed bad, and otherwise placing the returned block in its slot and zeroing the blocks of the other instances.

Second, Search takes the caller's private state as input, so different parties may find different witnesses. This does not matter: the analysis in Section 2.2 uses only whether a witness exists, which on the good event is the same for all honest parties, together with the fact that every witness decodes to the same sharing-fixed mask. The invariants and the fixed-point argument go through verbatim. Each of the at most $n+1$ passes makes $O(n^2)$ closure calls and $O(n)$ recovery calls, so $O(n^3)$ calls in all, hence the VSS inherits polynomial local computation from the kuWSS with (c).

Why our WSS satisfies (c). For the WSS of Section 2.3 the search turns out to be easy, for two reasons. The first is the form of the public checks performed by Out. For every proposed key assignment and opening transcript, Out returns $\perp$ exactly when one of the checks (G1)–(G4) in Step 2 of Protocol 7 fails. These checks depend only on the sharing transcript and the proposed keys, and separate into small constraint groups: a key-independent check of the repairs, one constraint per pair $\{i, j\}$ relating the opened overlaps S_{ij} and S_{ji}, and one constraint per pad copy Q_{ij} on its opened value.

The second is that a free key is, essentially, a free vote: The opened value of an EARLY label depends on the free keys only through *how many* of them accept its candidate, and for a label with a valid candidate P and checkpoint (d, T) one can write down, for any point α, a key that accepts, $(\alpha, P(\alpha), 0)$, and a key that rejects, $(\alpha,\ P(\alpha)+1,\ T(\alpha) - d(P(\alpha)+1))$. Hence if v_0 of the fixed keys accept and z free parties have well-formed deposits, the label can be made to open to its candidate iff $v_0 + z \geq t + 1$ and to $\perp$ iff $v_0 \leq t$, with explicit witnesses; labels that were replaced, that were never *activated* (neither required to be opened by the round-two data nor opened voluntarily by the holder, so that the decoder assigns them no value), or that were opened with an invalid candidate have a single possible outcome.

A constraint of the predicate involves at most two labels: the pair constraint for $\{i, j\}$ reads the opened values of S_{ij} and S_{ji}, and the constraint for Q_{ij} reads only Q_{ij}. Since each label has at most two attainable values, a constraint has at most four combinations of label values, its *outcome patterns*, and Search can simply test each of them. It remains to see that the patterns chosen for different constraints can be realized at the same time. A free party's key vector has one slot per label, and the value of a label is determined by the votes in its own slots alone. Realizing a chosen value for S_{ij} therefore only fixes the S_{ij} slots of the free parties, which no other constraint reads; and since no two constraints share a label, the slot assignments chosen for different constraints are assignments to disjoint parts of the free key vectors and can be combined freely. Thus $\mathsf{Search}(\cdot, \mathsf{acc})$ picks a satisfying pattern for every constraint and writes the corresponding slots, $\mathsf{Search}(\cdot, \mathsf{rej})$ picks a violating pattern for one constraint and fills the remaining slots arbitrarily, and both return none precisely when no pattern qualifies.

A call touches $O(n^2)$ labels and does polynomial-time field arithmetic; for the mask instances, whose inputs have length $m = O(n^3)$, a crude bound for all $O(n^3)$ calls is $O(n^{11})$ field operations. Section 7.4 and Appendix A.4 give the details.

2.5 Privacy of the composition

The proof is a coupling that changes the root secret by Δ while fixing the adversary's view. Shift the root polynomial by $\Delta\, p(X)p(Y)$, where p vanishes at the corrupt nodes and $p(0) = 1$, and compensate the honest pads and the IC polynomials exactly as in the privacy proof of the standalone WSS. Every honest key then moves by an $\mathbb{F}$-linear function of the shifts, with coefficients in $\mathbb{F}$ determined by the fixed checking points, public evaluation nodes, and challenges. To keep every masked broadcast C_i fixed, the honest masks must move as well, by a vector u solving

$$(\lambda I + A)\, u = -b_\Delta, \qquad A \in \mathbb{F}^{r \times r},\ r \leq nm,$$

where A records how mask shifts feed back into the keys of the mask instances. This is where the field extension enters: all nodes, checking points and challenges lie in a base field $\mathbb{F}$, and $\mathbb{E} = \mathbb{F}[\lambda]$ with λ of algebraic degree greater than nm, so $\det(\lambda I + A)$, a monic polynomial of degree r in λ over $\mathbb{F}$, cannot vanish. The compensation thus exists for every execution, the map is a bijection of the honest coins, and the adversary's view is identical for s and $s + \Delta$.

The same coupling, applied to one key instead of the input, is what makes the IC guarantees survive the composition. Keys for EARLY labels are broadcast in masked form in sharing round two, that is, *before* the round-three candidates are fixed, so observation (b) of Section 2.3 does not apply as stated. Switching the point of one honest key and compensating through the masks shows that the point remains uniform given the adversary's whole sharing view; unforgeability then holds for the actual round-three candidates with the standalone bound. Keys for LATE labels are never broadcast. A union bound over labels and honest parties gives the event on which all $n + 1$ instances are key-uniform, which is all the compiler needs. The full proofs are in Section 8.3 and Appendix B.1.

3 Related work

Round complexity of VSS. Gennaro, Ishai, Kushilevitz and Rabin [GIKR01] initiated the study of the round complexity of VSS, proving that three sharing rounds are necessary for perfect VSS with $t < n/3$ and that two rounds suffice for $t < n/4$, and Fitzi, Garay, Gollakota, Pandu Rangan and Srinathan [FGG+06] gave an efficient three-round perfect protocol with one reconstruction round. Katz, Koo and Kumaresan [KKK08] studied the round complexity when broadcast must be emulated over point-to-point channels. For statistical security, Patra, Choudhary, Rabin and Rangan [PCRR09] gave an efficient protocol with two sharing rounds and two reconstruction rounds for $t < n/3$, together with matching impossibility results for the sharing phase. Applebaum, Kachlon and Patra [AKP20a] subsequently obtained two-round statistical VSS at the same threshold supporting second-level sharing. For the optimal threshold $t < n/2$, Patra, Choudhary, Rabin and Rangan [PCRR09] showed that three sharing rounds are necessary, and Kumaresan, Patra and Rangan [KPR10] showed sufficiency, with an inefficient $(3, 2)$-round protocol. Applebaum, Kachlon and Patra [AKP23] obtained the first $(3, 1)$-round protocol, with complexity exponential in n. In the computational setting, Backes, Kate and Patra [BKP11] characterized the round complexity of VSS under standard assumptions, where non-interactive commitments make one-round reconstruction straightforward [Fel87, Ped92].

Round complexity of MPC. Round complexity has been a central measure for secure computation since Beaver, Micali and Rogaway [BMR90]. In the information-theoretic honest-majority setting, Applebaum, Brakerski and Tsabary [ABT18, ABT19] and Garg, Ishai and Srinivasan [GIS18] gave round-efficient protocols for sub-optimal thresholds, Ananth, Choudhuri, Goel and Jain [ACGJ18] studied the honest-majority computational case, and Applebaum, Kachlon and Patra settled the round complexity of perfectly secure MPC with $t < n/3$ at four rounds [AKP20b] and of statistically secure MPC with $t < n/2$, also at four rounds [AKP23]. The latter is built on their $(3, 1)$-round statistical VSS, and inherits its exponential complexity; our VSS is a natural starting point for an efficient version.

Information checking. Information checking, introduced by Rabin and Ben-Or [RB89] and made efficient and linear by Cramer, Damgård, Dziembowski, Hirt and Rabin [CDD+99], is the standard information-theoretic substitute for a signature with an honest majority. Its verifiers hold private keys and reach their verdicts locally, so agreement on an opening requires a voting round or, as in [AKP23], a committee-based signature whose opening has a common verdict.

4 Preliminaries

We now discuss our model and key buildings blocks used in our construction.

Model. We assume n parties $P_1, \ldots, P_n$, with a distinguished dealer D. The parties are connected via private authenticated channels and ideal broadcast. The adversary is rushing, static, computationally unbounded and controls the parties indexed by a fixed set $\mathcal{C} \subseteq [n]$ of size at most t, where $t < n/2$. We use $\mathcal{H} = [n] \setminus \mathcal{C}$ to denote the set of honest-party indices.

Inputs, states, and outputs. For a finite field $\mathbb{E}$ and a positive integer h, the dealer's input is a vector $x \in \mathbb{E}^h$. A sharing scheme consists of two phases: During the *Sharing phase* the dealer distributes its input, and during the *Reconstruction phase* every party later outputs an element of $\mathbb{E}^h \cup \{\perp\}$.

Definition 4.1 (Statistical weak secret sharing)**.** *A WSS scheme with error ε satisfies the following properties against every static rushing adversary corrupting a set $\mathcal{C}$ of at most t parties. Let S denote the honest parties' joint view at the end of sharing, including their local states and the public sharing transcript.*

(i) ***Correctness.*** *If the dealer is honest with input $x \in \mathbb{E}^h$, then with probability at least $1 - \varepsilon$, the value $\rho(S)$ specified in (iii) equals x and every honest party outputs x at the end of reconstruction.*

(ii) ***Perfect privacy.*** *If the dealer is honest, then for every pair of inputs $x_0, x_1 \in \mathbb{E}^h$, the adversary's views at the end of sharing are identically distributed.*

(iii) ***Weak commitment.*** *There exists a deterministic function ρ of S, taking values in $\mathbb{E}^h$ and fixed independently of the adversary's reconstruction strategy, such that for every such strategy, with probability at least $1 - \varepsilon$, all honest parties output the same value in $\{\rho(S), \perp\}$.*

Definition 4.2 (Statistical verifiable secret sharing)**.** *A VSS scheme with error ε satisfies the following properties against every static rushing adversary corrupting a set $\mathcal{C}$ of at most t parties. Let S denote the honest parties' joint view at the end of sharing, including their local states and the public sharing transcript.*

(i) ***Correctness.*** *If the dealer is honest with input $x \in \mathbb{E}^h$, then with probability at least $1 - \varepsilon$, the value $\rho(S)$ specified in (iii) equals x and every honest party outputs x at the end of reconstruction.*

(ii) ***Perfect privacy.*** *If the dealer is honest, then for every pair of inputs $x_0, x_1 \in \mathbb{E}^h$, the adversary's views at the end of sharing are identically distributed.*

(iii) ***Commitment.*** *There exists a deterministic function ρ of S, taking values in $\mathbb{E}^h \cup \{\perp\}$ and fixed independently of the adversary's reconstruction strategy, such that for every such strategy, with probability at least $1 - \varepsilon$, all honest parties output $\rho(S)$.*

5 Information checking

Information checking (IC) authenticates a sender's message to a holder in the presence of verifiers holding private keys. Our formulation is the standard one, with one presentational change: the

verifier's check is an explicit public algorithm Open of the candidate and the key, because the protocols that use IC will evaluate it with keys other than the evaluating party's own, including keys published after the candidate they judge has been fixed.

Definition 5.1 (IC syntax)**.** *An IC protocol* IC *is run by the parties* $P_1, \dots, P_n$ *with a designated* sender S *and a designated* holder $H \neq S$*; every party, including* S *and* H*, acts as a verifier. It consists of three subprotocols* $\mathsf{Distribute}$*,* Verify*,* Reveal*:*

(i) $\mathsf{Distribute}(x)$*, one round. The sender, on input* $x \in \mathbb{E}^h$*, sends private messages. Outputs: the holder obtains* $x_H \in \mathbb{E}^h \cup \{\perp\}$ *and a witness* obj*; each party* P_w *obtains a key* key_w*.*

(ii) Verify*, two broadcast rounds. The holder broadcasts a* checkpoint cp*; the sender then broadcasts a* finish*. Public output:* $\mathrm{out} \in \{\mathsf{ok}\} \cup \{\mathsf{fail}(x') : x' \in \mathbb{E}^h\}$.

(iii) Reveal*, one broadcast round, which may coincide with the second round of* Verify*. The holder broadcasts a candidate* (x', obj')*; an honest holder sends* (x_H, obj)*. Each party* P_w *outputs* x' *if* $\mathsf{Open}(x', \mathrm{obj}', \mathrm{cp}, \mathrm{key}_w) = 1$ *and* $\perp$ *otherwise, where* Open *is a public deterministic algorithm defined on all strings.*

An output $\mathsf{fail}(x')$ *of* Verify *publicly replaces the message by* x' *and overrides* Reveal*.*

Definition 5.2 (IC security)**.** *We consider the following properties.*

(P1) ***Correctness.*** *If sender and holder are honest, then it holds that* $\mathrm{out} = \mathsf{ok}$*,* $x_H = x$*, and* $\mathsf{Open}(x, \mathrm{obj}, \mathrm{cp}, \mathrm{key}_w) = 1$ *for every* w*.*

(P2) ***Honest failure.*** *If the sender is honest, then* $\mathrm{out} \in \{\mathsf{ok}, \mathsf{fail}(x)\}$*.*

(P3) ***Nonrepudiation.*** *Let the holder be honest with* $x_H \neq \perp$*. For every verifier* P_w *and every key string* key_w *that is a function of the Distribute messages,*

$$\Pr\big[\mathsf{Open}(x_H, \mathrm{obj}, \mathrm{cp}, \mathrm{key}_w) = 0\big] \leq \varepsilon_{\mathrm{nr}},$$

over the holder's Verify coins. Sender and verifier may be corrupt and the key string malformed.

(P4) ***Unforgeability.*** *Let the sender be honest and the holder corrupt. For every honest verifier* P_w*, an adversary that receives all Distribute messages except* key_w*, plays the holder in Verify, and fixes a candidate* (x', obj') *satisfies*

$$\Pr\big[\mathrm{out} = \mathsf{ok} \ \wedge \ x' \neq x \ \wedge \ \mathsf{Open}(x', \mathrm{obj}', \mathrm{cp}, \mathrm{key}_w) = 1\big] \leq \varepsilon_{\mathrm{au}}.$$

Since the candidate is fixed before key_w *is involved, the bound is unaffected if* key_w *is disclosed afterwards; the protocols below rely on this.*

(P5) ***Privacy.*** *If sender and holder are honest, then for every* $V \subseteq [n]$ *with* $|V| \leq t$ *the distribution of* $(\mathrm{cp}, \mathrm{out}, (\mathrm{key}_w)_{w \in V})$ *is the same for all inputs.*

We write ε *for a common bound on* $\varepsilon_{\mathrm{nr}}$ *and* $\varepsilon_{\mathrm{au}}$*.*

5.1 A packed realization

We give a polynomial-time realization of IC with message space $\mathbb{E}^h$. The witness is a polynomial, $\mathrm{obj} = F$; a candidate witness is written P. The holder keeps (F, G) as private state for computing its checkpoint.

Parameters and validity. Choose distinct packing nodes $\beta_1, \dots, \beta_h \in \mathbb{F}$, with $\beta_1 = 0$ when $h = 1$. Put

$$\delta = h + t - 1, \qquad \mathcal{A} = \mathbb{F} \setminus \{0, \beta_1, \dots, \beta_h\} \neq \varnothing.$$

Polynomials belong to $\mathbb{E}[X]$ and have degree at most δ; write $\mathsf{val}(F) = (F(\beta_1), \dots, F(\beta_h))$. A *valid key* is a triple in $\mathcal{K} = \mathcal{A} \times \mathbb{E} \times \mathbb{E}$; a *valid checkpoint* is a pair (d, T) with $d \in \mathbb{F}^\times$ and $\deg T \leq \delta$; a *valid candidate* is a pair (x', P) with $x' \in \mathbb{E}^h$, $\deg P \leq \delta$ and $\mathsf{val}(P) = x'$. Fix $\alpha_0 \in \mathcal{A}$ and let $\mathsf{Norm}(k) = k$ for $k \in \mathcal{K}$ and $\mathsf{Norm}(k) = (\alpha_0, 0, 0)$ otherwise. Every valid key is three elements of $\mathbb{E}$, independently of h.

The verification algorithm. For a candidate (x', P), a checkpoint cp and a key string k, let $\mathsf{Open}(x', P, \mathrm{cp}, k) = 0$ unless (x', P) and $\mathrm{cp} = (d, T)$ are valid, in which case, with $(\alpha, y, z) = \mathsf{Norm}(k)$,

$$\mathsf{Open}(x', P, \mathrm{cp}, k) = [P(\alpha) = y] \ \vee \ [T(\alpha) \neq dy + z].$$

The first disjunct checks the candidate against the key; the second excuses the holder when the key is inconsistent with the public checkpoint.

Protocol 3: Packed IC

$\mathsf{Distribute}(x)$ — **one round of private messages.**

Sender	On input $x \in \mathbb{E}^h$, sample independent uniform $F, G \in \mathbb{E}[X]$ of degree at most δ with $\mathsf{val}(F) = x$. Send (F, G) to the holder. For every party P_w, sample $\alpha_w \leftarrow \mathcal{A}$ and send $\mathrm{key}_w = (\alpha_w, F(\alpha_w), G(\alpha_w))$ to P_w.
Holder	Output $x_H = \mathsf{val}(F)$ and $\mathrm{obj} = F$; keep (F, G).
Party P_w	Output key_w as received.

Verify — **two broadcast rounds.**

Holder	(round 1) Sample $d \leftarrow \mathbb{F}^\times$ and broadcast the checkpoint $\mathrm{cp} = (d, T)$, $T = dF + G$.
Sender	(round 2) Broadcast the finish ok if $\mathrm{cp} = (d, T)$ is valid and $T = dF + G$ as polynomials, and $\mathsf{fail}(x)$ otherwise.
Output	out is the finish; a missing or invalid finish is read as ok.

Reveal — **one broadcast round.**

Holder	Broadcast the candidate (x_H, F).
Party P_w	Output x' if $\mathsf{Open}(x', P, \mathrm{cp}, \mathrm{key}_w) = 1$ for the broadcast candidate (x', P), and $\perp$ otherwise.

Malformed data. If the holder receives an invalid pair in Distribute, it outputs $x_H = \perp$, broadcasts an invalid checkpoint in Verify and an invalid candidate in Reveal, and every party then outputs $\perp$ in Reveal. Reveal may coincide with the second Verify round, since the candidate depends only on the holder's Distribute output; an output $\mathsf{fail}(x')$ of Verify overrides the Reveal outputs, as in Definition 5.1.

Lemma 5.3 (Security of packed IC)**.** *The protocol satisfies correctness, honest failure, and privacy perfectly; nonrepudiation with* $\varepsilon_{\mathrm{nr}} = 1/(|\mathbb{F}| - 1)$*; and unforgeability with* $\varepsilon_{\mathrm{au}} = \delta/|\mathcal{A}|$*, per verifier*

and actual candidate. Hence one may take

$$\varepsilon = \max\Big\{\frac{1}{|\mathbb{F}|-1}, \frac{\delta}{|\mathcal{A}|}\Big\} = \frac{\delta}{|\mathcal{A}|} \quad (t \geq 1).$$

The unforgeability bound also holds in any execution in which the target point α_w is uniform on $\mathcal{A}$ conditioned on the adversary's view when it fixes its candidate and on (F, G); under that hypothesis a candidate $P \neq F$ with $\mathsf{val}(P) = x$ also passes with probability at most $\delta/|\mathcal{A}|$.

Proof. (P1): with honest endpoints the checkpoint is $T = dF + G$, so the finish is ok, $x_H = x$, and every supplied key $(\alpha_w, F(\alpha_w), G(\alpha_w))$ accepts the candidate (x, F) through the first disjunct. (P2): an honest sender announces only ok or $\mathsf{fail}(x)$.

(P3): fix the holder's valid state and any key string k received by any party, both determined by the Distribute messages. The normalized tuple $(\alpha, y, z) = \mathsf{Norm}(k)$ is fixed before the holder samples d. Rejection of the candidate (x_H, F) requires $F(\alpha) \neq y$ and $d\,(F(\alpha) - y) = z - G(\alpha)$, which holds for at most one $d \in \mathbb{F}^{\times}$; hence the probability over d is at most $1/(|\mathbb{F}| - 1)$. The argument does not use the finish and applies to a corrupt sender or verifier and to malformed k.

(P4): all sender-supplied keys are valid. The target point α_w is uniform on $\mathcal{A}$ and independent of every other Distribute message, including F, G and the other parties' keys, so the adversary's checkpoint and candidate are independent of it; the honest sender's finish is a polynomial identity test that does not involve it. If out $= \mathsf{ok}$ then $T = dF + G$, so at the target key the second disjunct is false, and a valid candidate (x', P) is accepted only if $(P - F)(\alpha_w) = 0$. Since $x' \neq x$ gives $P \neq F$, with $\deg(P - F) \leq \delta$, this has at most δ solutions, hence probability at most $\delta/|\mathcal{A}|$. The same count applies to any $P \neq F$ under the conditional-uniformity hypothesis.

(P5): fix a set V of at most t parties and the distinct points $\Gamma \subseteq \mathcal{A}$ of their keys. For every $\xi \in \mathbb{E}^h$ there is a polynomial H of degree at most $h + t - 1 = \delta$ with $\mathsf{val}(H) = \xi$ and $H|_{\Gamma} = 0$. For each fixed d the map $(F, G) \mapsto (F + H, G - dH)$ is a bijection of the sender's polynomial spaces that changes the message by ξ and preserves cp, every key in V, and out $= \mathsf{ok}$. Averaging over the points and d, the distribution of $(\mathrm{cp}, \mathrm{out}, (\mathrm{key}_w)_{w \in V})$ does not depend on x. □

6 WSS with explicit verification keys

We consider WSS schemes whose reconstruction consists of one broadcast round followed by a local computation, part of each broadcast being a vector of *verification keys*. For such schemes we define two strengthenings of weak commitment: the guarantee must hold for every assignment of corrupt keys simultaneously, and the decision to reject must depend on the sharing transcript and the keys alone.

Definition 6.1 (Reconstruction type)**.** *A WSS scheme is* of the broadcast-then-decode type with key length ℓ *if its reconstruction phase has the following form.*

1. *Every party P_j broadcasts a message $o_j = (\omega_j, \kappa_j)$ computed from its sharing state σ_j, before receiving any reconstruction message: an* opening ω_j*, and a* verification vector $\kappa_j \in \mathbb{E}^{\ell}$ *that is already determined by P_j's state at the end of sharing round one. The length ℓ does not depend on the secret length h.*

2. *Every party P_i then outputs*

$$\mathsf{Out}(\sigma_i, K; \tau), \qquad K = (k_j)_{j \in [n]}, \quad \tau = (\omega_j)_{j \in [n]},$$

where Out *is a public deterministic polynomial-time function,* K *collects the broadcast verification vectors and* τ *the broadcast openings,* σ_i *contains the public sharing transcript* Γ_{share}*, and a missing or malformed message part is read as* bad*.*

Call K *honest-consistent* if $k_j = \kappa_j$ for every honest P_j; its corrupt coordinates are arbitrary elements of $\mathbb{E}^\ell \cup \{\mathsf{bad}\}$. Similarly, an opening vector $\tau = (w_j)_{j\in[n]}$ is *honest-consistent* if $w_j = \omega_j$ for every honest P_j; its corrupt coordinates are arbitrary strings.

Definition 6.2 (Key-uniform WSS)**.** *A WSS scheme of the broadcast-then-decode type is* key-uniform *with error* ε *if, with* ρ *the value of Definition 4.1 and* τ *the openings actually broadcast, except with probability* ε*, the following properties hold simultaneously for all honest-consistent* K*:*

(a) ***Uniform weak commitment.*** *All honest* P_i *obtain the same* $\mathsf{Out}(\sigma_i, K; \tau) \in \{\rho, \perp\}$*; if the dealer is honest with input* x*, it equals* x*.*

(b) ***Sharing-determined rejection.*** *For every honest* P_i *and all honest-consistent* τ_1, τ_2*,* $\mathsf{Out}(\sigma_i, K; \tau_1) = \perp \Leftrightarrow \mathsf{Out}(\sigma_i, K; \tau_2) = \perp$*.*

Together, (a) and (b) say that on the good event the rejection decision for an honest-consistent K is a function of K and the honest parties' views at the end of sharing: it is the same at every honest party and cannot be influenced by the corrupt parties' openings. A scheme may satisfy (b) in the stronger, structural form in which this decision is a public function $\mathsf{Reject}(\Gamma_{\text{share}}, K)$ of the sharing transcript and K, for every K.

Remark 6.3. (a) differs from Definition 4.1 only in the placement of a quantifier: one event covers every honest-consistent K at once, rather than one bound per adversarial choice of corrupt keys. Uniformity is asked of the keys and not of the openings because the latter would be unsatisfiable: an opening chosen with knowledge of a hidden verification point always passes somewhere, whereas keys only count votes on openings fixed during sharing and so can toggle rejection but not forge.

7 A key-uniform WSS protocol

We construct a key-uniform WSS scheme (Definition 6.2) of broadcast-then-decode type with three sharing rounds and one reconstruction round. Section 7.1 explains the idea, Section 7.2 gives the protocol, and Section 7.3 proves the properties of Definitions 4.1 and 6.2 one by one; the longer proofs are in Appendix A.

7.1 Overview

Rows. The dealer D chooses a symmetric bivariate polynomial B of degree t with $B(0,0) = s$ and gives every other party P_i its row $B(i,\cdot)$. Any t rows reveal nothing about s, any $t+1$ consistent rows determine B, and two rows overlap at one value, $B(i,j) = B(j,i)$. If the dealer is corrupt, at least $n - t \geq t+1$ row holders are honest, so the honest rows alone determine the secret; the protocol must ensure that they are mutually consistent, or else reject the dealer, and that at reconstruction no corrupt row inconsistent with them is used.

If the dealer is honest there may be only t honest row holders; authentication then guarantees that every retained row is a true row, and the dealer's fallback supplies the input when fewer than $t+1$ rows are retained.

Pads. To compare their overlap without revealing it, P_i sends P_j a random pad r_{ij} in round one, and in round two both broadcast the overlap plus the pad: $a_{ij} = B(i,j) + r_{ij}$ from P_i and $b_{ji} = B(j,i) + r_{ij}$ from P_j. The dealer, who also receives a copy of every pad, broadcasts its own versions. A mismatch flags a *dispute* between P_i and P_j, or a *trigger* on row i if P_i's broadcast contradicts the dealer's. In round three the dealer publicly repairs triggered rows, and disputed overlaps are opened.

Authentication. Every value that may later have to be defended is authenticated with a IC instance (Section 5), identified by a *label*; for $i \neq j \in I$ and $\ell \in \{0, \ldots, t\}$, where $\gamma_{i,\ell}$ are the coefficients of row i:

Label	Sender $\to$ holder	Message	Mode
$W_{i,\ell}$	$D \to P_i$	$\gamma_{i,\ell}$, coefficient of row i	LATE
S_{ij}	$D \to P_i$	$B(i,j)$, row intersection	EARLY
Q_{ij}	$P_i \to D$	r_{ij}, dealer's pad copy	EARLY
R_{ij}	$P_i \to P_j$	r_{ij}, neighbour's pad copy	LATE

An instance thus has

$$M = (n-1)(t+1) + 3(n-1)(n-2) < 4n^2$$

labels. A label is *revealed* in one of two modes. EARLY labels (S, Q) are revealed in sharing round three when a dispute or repair requires it, and their keys are *exported*: the verification vector κ_w of Definition 6.1 is P_w's keys for all S and Q labels. At reconstruction these keys are available to everyone, so an EARLY opening is accepted by a public vote of all n keys with threshold $t+1$, the same at every party. LATE labels (W, R) are revealed in the reconstruction round and their keys stay private, so each party judges them with its own key; their results only decide whether an individual row is used.

Reconstruction. Each party publishes its row (through the W labels), its incoming pads (through the R labels) and its exported keys. Decoding has two stages. A public *rejection test*, which reads only the sharing transcript and the keys, rejects the dealer if the opened overlaps and the repaired rows contradict one another. If it passes, each party keeps the rows that are consistent with their neighbours, as witnessed by opened pads and overlaps, and interpolates the first $t+1$ of them. The analysis shows that every honest row is kept and every kept row lies on the polynomial through the honest rows, so all honest parties obtain the same value, whichever corrupt rows they happen to keep.

7.2 Protocol

Notation. The dealer D holds $s \in \mathbb{E}^h$; the other $n-1$ parties, the *row holders*, are indexed by I, and $i, j, k \in I$ are distinct unless stated otherwise; ℓ ranges over $\{0, \ldots, t\}$. A value is *finite* if it lies in $\mathbb{E}^h$. Party indices are distinct nonzero elements a_i of $\mathbb{F} \subseteq \mathbb{E}$, and evaluation at i means evaluation at a_i; vector arithmetic is coordinatewise. A missing or malformed message component reads as $\perp$ if it is a value and as 1 if it is a flag, and a comparison with a $\perp$ operand fails.

Each label L of the table is a IC instance with the listed sender S_L, holder H_L and message; all n parties are verifiers of every instance, and IC Distribute and the two Verify rounds run in the three sharing rounds. $\mathsf{IC.Distribute}_L$, $\mathsf{IC.Verify}_L$ and $\mathsf{IC.Reveal}_L$ denote the sender's Distribute step,

the holder's checkpoint step and the holder's Reveal broadcast; the sender's finish is computed from its state and the checkpoint cp_L. For a holder P_v we write $x_v(L)$ and $\omega_v(L)$ for its Distribute outputs, and $K_{L,w}$ for verifier P_w's stored (normalized) key. A holder with $x_v(L) = \perp$ sends an invalid candidate when asked to reveal. Let $c_L = 1$ exactly when the Verify output of L is a valid replacement $\mathsf{fail}(x_L^{\mathrm{rep}})$.

Verification vectors. As a verifier, every party P_w receives one key per label in round one and stores it normalized, so every stored key $K_{L,w}$ is valid (Section 5.1); Open applies the same normalization to any key string. The *verification vector* κ_w of P_w is the concatenation, in a fixed public order, of its stored keys for the S and Q labels only:

$$\kappa_w \in \mathbb{E}^{m_{\mathrm{WS}}}, \qquad m_{\mathrm{WS}} = 6(n-1)(n-2),$$

three field elements per key. It is fixed at the end of round one, is independent of h, and stays private during sharing. The keys for the W and R labels never leave their verifier.

Proposed assignments and votes. The decoder is evaluated on a *proposed assignment* $k = (k_1, \ldots, k_n)$, one entry per party, where k_w is either a vector in $\mathbb{E}^{m_{\mathrm{WS}}}$ or the symbol bad. We write $k_w[L]$ for the entry of k_w in the slot of label L; every slot of bad is bad. The vote of entry $k_w[L]$ on a candidate (x', ω') of label L is

$$\mathsf{Check}_L(x', \omega'; k_w[L]) = \begin{cases} \mathsf{Open}(x', \omega', \mathrm{cp}_L, k_w[L]), & k_w[L] \in \mathcal{K}_L, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

For a stored original key, Check_L and Open agree.

Protocol 4: Sharing round 1

Input. The dealer D holds $s \in \mathbb{E}^h$.

1. Dealer D samples B uniformly subject to

$$B(X, Y) = B(Y, X), \qquad \deg_X B, \deg_Y B \le t, \qquad B(0, 0) = s,$$

and writes $B(i, Y) = \sum_{\ell=0}^{t} \gamma_{i,\ell} Y^\ell$. Each P_i samples independent pads $r_{ij} \leftarrow \mathbb{E}^h$.

2. For every label L of the table in Section 7.1, its sender runs $\mathsf{IC.Distribute}_L$ on the listed message: D sends $\gamma_{i,\ell}$ under $W_{i,\ell}$ and $B(i, j)$ under S_{ij} to P_i; P_i sends r_{ij} under Q_{ij} to D and under R_{ij} to P_j. Holders obtain $x_v(L), \omega_v(L)$ and every party P_w its keys $K_{L,w}$, from which it forms κ_w.

3. Each row holder P_i sets

$$g_i(Y) = \begin{cases} \displaystyle\sum_{\ell=0}^{t} x_i(W_{i,\ell}) Y^\ell, & \text{all } x_i(W_{i,\ell}) \text{ finite}, \\ 0, & \text{otherwise}, \end{cases}$$

and $\mathrm{inv}_i = 1$ if some $x_i(W_{i,\ell})$ or $x_i(S_{ij})$ is $\perp$ or some $x_i(S_{ij}) \neq g_i(j)$, else $\mathrm{inv}_i = 0$. It records its incoming pad $r_{ji}^{(i)} = x_i(R_{ji})$ and the flag $\phi_{ij} = [r_{ji}^{(i)} = \perp]$. The dealer records $r_{ij}^D = x_D(Q_{ij})$.

Protocol 5: Sharing round 2

1. For every label L, holder H_L runs $\mathsf{IC.Verify}_L$, broadcasting the checkpoint cp_L.
2. Each P_i broadcasts inv_i, the flags ϕ_{ij}, and

$$a_{ij} = g_i(j) + r_{ij}, \qquad b_{ij} = \begin{cases} g_i(j) + r^{(i)}_{ji}, & \phi_{ij} = 0, \\ \perp, & \phi_{ij} = 1. \end{cases}$$

   The dealer broadcasts $a^D_{ij} = B(i,j) + r^D_{ij}$ and $b^D_{ij} = B(i,j) + r^D_{ji}$.
3. Every party first *normalizes* the received comparisons: if the received flag ϕ_{ij} is 1 (a missing or malformed flag reads as 1), it replaces the received b_{ij} by $\perp$, whatever value P_i broadcast. All later uses of b_{ij}, in this round and in the decoder, refer to the normalized value. It then computes the public *row trigger* η_i (row i must be repaired) and the *dispute flag* p_{ij}:

$$\begin{aligned} \eta_i &= \mathrm{inv}_i \ \vee \bigvee_{j \in I \setminus \{i\}} [a_{ij} \neq a^D_{ij}], \\ p_{ij} &= [a_{ij} \neq b_{ji}] \vee [a_{ji} \neq b_{ij}] \vee [a_{ij} \neq a^D_{ij}] \vee [b_{ij} \neq b^D_{ij}]. \end{aligned} \tag{5}$$

4. Each sender S_L computes its finish for L from its state and cp_L; let $\widehat{c}_L = 1$ if it is $\mathsf{fail}(x^{\mathrm{rep}}_L)$ and $\widehat{c}_L = 0$ otherwise. The dealer sets

$$U^{\mathrm{plan}} = \Big\{ i \in I : \ \eta_i \vee \bigvee_{j \neq i} \widehat{c}_{S_{ij}} \vee \bigvee_{\ell} \widehat{c}_{W_{i,\ell}} \Big\}.$$

Protocol 6: Sharing round 3

1. For every label L, sender S_L broadcasts its finish (ok if $\widehat{c}_L = 0$, $\mathsf{fail}(x^{\mathrm{rep}}_L)$ if $\widehat{c}_L = 1$).
2. The dealer broadcasts $U = U^{\mathrm{plan}}$ and $u_i(Y) = B(i,Y)$ for every $i \in U$.
3. Row holder P_i runs $\mathsf{IC.Reveal}_{S_{ij}}$, broadcasting $(x_i(S_{ij}), \omega_i(S_{ij}))$, if $p_{ij} \vee \widehat{c}_{R_{ij}} \vee \bigvee_{k \neq i} \widehat{c}_{Q_{ik}} = 1$.
4. The dealer runs $\mathsf{IC.Reveal}_{Q_{ij}}$, broadcasting $(x_D(Q_{ij}), \omega_D(Q_{ij}))$, if $i \notin U^{\mathrm{plan}}$ and $j \in U^{\mathrm{plan}}$.

Required and activated labels. After round three, let U be the dealer's declared repair set and let $c_L = 1$ exactly when the Verify output of L is a valid replacement $\mathsf{fail}(x^{\mathrm{rep}}_L)$. An EARLY label is *required* to be opened when

$$\mathrm{req}(S_{ij}) = p_{ij} \vee c_{R_{ij}} \vee \bigvee_{k \in I \setminus \{i\}} c_{Q_{ik}}, \qquad \mathrm{req}(Q_{ij}) = [i \notin U] \wedge [j \in U]; \tag{6}$$

these are exactly the conditions under which an honest holder sends its candidate in round three, evaluated on the actual round-three flags. A label is *activated* if it is required or if its holder broadcast any candidate, valid or not. A replacement overrides the opening and need not activate the label.

Protocol 7: Reconstruction: one round and local decoding

Broadcast. Each party P_w broadcasts κ_w. Each row holder P_i runs $\mathsf{IC.Reveal}_L$ for every $L = W_{i,\ell}$ and for every $L = R_{ji}$ with $c_{R_{ji}} = 0$ and $\phi_{ij} = 0$. The dealer broadcasts its input as fallback. The announced vectors form the assignment k^{ann} (a malformed announcement reads as bad); the candidates and the fallback form the opening vector τ.
Decoding at P_h: $\mathsf{Dec}_{O,h}(\sigma_h, k; \tau)$, run with $k = k^{\mathrm{ann}}$.

1. *Values of labels.* For a label L with broadcast candidate (x', ω'): if $c_L = 1$ the value is x_L^{rep}; if no valid candidate was broadcast it is $\perp$; otherwise it is x' if the candidate is accepted and $\perp$ if not, where an EARLY candidate is accepted iff $\sum_{w=1}^{n} \mathsf{Check}_L(x', \omega'; k_w[L]) \geq t+1$ (all n parties vote) and a LATE candidate iff Open accepts it under P_h's stored key $K_{L,h}$. Write $\mathsf{Early}(L; k)$ and $\mathsf{Late}_h(L)$ for these values, with $\mathsf{Early}(L; k) = \mathsf{unavailable}$ for a non-activated label, and set

$$s_{ij} = \mathsf{Early}(S_{ij}; k), \qquad q_{ij} = \mathsf{Early}(Q_{ij}; k), \qquad w_{i,\ell} = \mathsf{Late}_h(W_{i,\ell}), \qquad v_{ij} = \mathsf{Late}_h(R_{ij}),$$

   except that $s_{ij} = u_i(j)$ whenever $i \in U$. A malformed repair declaration or a malformed u_i fails (G1); such u_i are not evaluated.

2. *Rejection.* Output $\perp$ if any of the following fails; this test, $\mathsf{Reject}_O(\Gamma_O, k)$, reads only Γ_O and k.
   - (G1) $U \subseteq I$, $|U| \leq t$, U contains every i with $\eta_i = 1$ or a replacement on some S_{ij} or $W_{i,\ell}$, every u_i has degree at most t, and u_i agrees with every replacement on its row ($u_i(j) = x^{\mathrm{rep}}_{S_{ij}}$, $[Y^\ell]u_i = x^{\mathrm{rep}}_{W_{i,\ell}}$).
   - (G2) For every pair $\{i, j\}$, the finite values among s_{ij}, s_{ji} agree.
   - (G3) Every activated Q_{ij} has finite q_{ij}.
   - (G4) For every $i \notin U$, $j \in U$ with $c_{Q_{ij}} = 0$: $a^D_{ij} = b^D_{ji} = u_j(i) + q_{ij}$, all operands finite.

3. *Rows.* Set $f_i = u_i$ for $i \in U$. For $i \notin U$ set $f_i = \sum_\ell w_{i,\ell} Y^\ell$ if all $w_{i,\ell}$ are finite, and *retain* row i iff all $w_{i,\ell}$ are finite, $s_{ij} = f_i(j)$ for every activated S_{ij}, and for every $j \neq i$:

$$\begin{aligned} j \in U &\;\Rightarrow\; f_i(j) = u_j(i), \\ v_{ij} \text{ finite} &\;\Rightarrow\; f_i(j) + v_{ij} = a_{ij}, \\ c_{R_{ji}} = 0 \text{ and } \phi_{ij} = 0 &\;\Rightarrow\; v_{ji} \text{ finite and } b_{ij} - v_{ji} = f_i(j). \end{aligned}$$

   Repaired rows are always retained.

4. *Output.* If at least $t+1$ rows are retained, let J be the first $t+1$ of them and output $\sum_{i \in J} L_i^J(0) f_i(0)$, where L_i^J is the Lagrange basis polynomial of a_i on $\{a_j : j \in J\}$; otherwise output the dealer's fallback if it is a vector in $\mathbb{E}^h$, and the zero vector of $\mathbb{E}^h$ if the fallback is missing or malformed.

This is a scheme of broadcast-then-decode type (Definition 6.1): κ_w is the verification vector, the candidates and the fallback are the opening ω_w, and $\mathsf{Dec}_{O,h}$ is the output function $\mathsf{Out}(\sigma_h, k; \tau)$.

7.3 Security

Theorem 7.1 (Key-uniform WSS)**.** *Let $t < n/2$. Against a static rushing adversary, the protocol of Section 7.2, instantiated with the IC of Section 5.1, is a key-uniform WSS scheme (Definitions 4.1 and 6.2) with three sharing rounds, one reconstruction round, key length m_{WS}, perfect sharing privacy, polynomial communication and computation, and error*

$$\varepsilon_{\mathrm{WSS}} = Mn\Big(\frac{1}{q-1} + \frac{\delta}{|\mathcal{A}|}\Big), \qquad q = |\mathbb{F}|,\ \delta = h + t - 1. \tag{7}$$

The proof is organized by property. Lemma 7.2 is privacy. Lemma 7.3 defines an event $\mathcal{G}$ of probability $1 - \varepsilon_{\mathrm{WSS}}$ on which the authentication of every label behaves. On $\mathcal{G}$, Lemma 7.4 is correctness, Lemma 7.5 is uniform weak commitment (property (a)), and Lemma 7.6 is sharing-determined rejection (property (b)). The weak commitment of Definition 4.1 is the special case of (a) in which k is the announced assignment.

Lemma 7.2 (Privacy)**.** *For an honest dealer, the adversary's view through the end of sharing has the same distribution for every input $s \in \mathbb{E}^h$.*

Proof sketch. Changing s to $s + \Delta$ is compensated by translating the dealer's polynomial by $\Delta\, p_{\mathcal{C}}(X) p_{\mathcal{C}}(Y)$, where $p_{\mathcal{C}}$ vanishes at the corrupt indices and $p_{\mathcal{C}}(0) = 1$; translating honest pads so that every public comparison is unchanged; and translating the IC polynomials of honest-sender, honest-holder labels so that every corrupt key and every checkpoint is unchanged. All messages the adversary sees are then identical, and the translation is a bijection of the honest coins. Appendix A.3 verifies this message by message, including the round-three openings. □

Lemma 7.3 (Authentication)**.** *Let $\mathcal{G}$ be the event that for every label L and every honest verifier P_w: (i) if the holder is honest with $x_{H_L}(L) \neq \perp$, its prescribed candidate is accepted by P_w's key; (ii) if the sender is honest and $c_L = 0$, every actual candidate different from the sender's witness is rejected by P_w's key. Then $\Pr[\mathcal{G}] \geq 1 - \varepsilon_{\mathrm{WSS}}$. On $\mathcal{G}$, an honest holder's candidate is accepted by at least $n - t \geq t + 1$ keys and a false candidate against an honest sender by at most t. Consequently, for every honest-consistent k: an activated EARLY label with an honest holder, $x_{H_L}(L) \neq \perp$ and no replacement opens to $x_{H_L}(L)$; an EARLY or LATE label with an honest sender opens to the sender's message, to its replacement (which carries the true message), or to $\perp$; and a LATE label with an honest holder, $x_{H_L}(L) \neq \perp$ and no replacement opens to $x_{H_L}(L)$ at every honest party.*

Proof sketch. (ii) is IC unforgeability: the candidate is fixed before the key is released (EARLY) or the key is never released (LATE), so the honest key's point is uniform given the candidate, and a wrong candidate passes with probability at most $\delta/|\mathcal{A}|$; Appendix A.1 proves the required point hiding. (i) is nonrepudiation, probability at most $1/(q-1)$ over the holder's challenge. A union bound over M labels and n verifiers gives the bound; the vote counts follow since at most t parties are corrupt. □

From now on fix an execution in $\mathcal{G}$. All remaining statements are deterministic and hold simultaneously for every honest-consistent k.

Lemma 7.4 (Correctness)**.** *If the dealer is honest, then $\mathsf{Reject}_O(\Gamma_O, k) = 0$ and every honest party outputs s.*

Proof. For honest P_i, its received row and intersections are correct, its pad copy at the owner is correct, and its owner-to-holder labels have no replacement. Hence $\eta_i = 0$ and $i \notin U$. Thus $U \subseteq \mathcal{C}$, and G1 holds with true repairs and true row replacements. Every finite s_{ij} is a true intersection by sender authenticity, giving G2. The only activated Q openings are the prescribed ones from the honest owner; their valid received payloads open reliably. G3 follows, and the owner's symmetric polynomial makes G4 an identity.

Every repaired row is true. A retained unrepaired row has finite coefficients from honest-sender W labels, so is also a true row. If at least $t+1$ rows remain they interpolate to the input. Otherwise the honest owner's fallback is the input. □

Lemma 7.5 (Uniform weak commitment)**.** *If the dealer is corrupt, there is $\rho_O \in \mathbb{E}^h$ determined by the honest parties' views at the end of sharing such that for every honest P_h, $\mathsf{Dec}_{O,h}(\sigma_h, k; \tau) = \perp$ if $\mathsf{Reject}_O(\Gamma_O, k) = 1$ and $\mathsf{Dec}_{O,h}(\sigma_h, k; \tau) = \rho_O$ otherwise.*

Proof sketch. Call the *prefix row* of honest P_i the repaired row u_i if $i \in U$ and its received row g_i otherwise; prefix rows are fixed by the honest views. Suppose $\mathsf{Reject}_O(\Gamma_O, k) = 0$. Appendix A.2 proves two facts: every honest row is retained by every honest party with its prefix row (Lemma A.3), and every row retained by any honest party agrees with every honest prefix row at their overlap (Lemma A.4). There are at least $n - t \geq t+1$ honest rows, so they determine a symmetric polynomial B_H whose row i is each honest prefix row; set $\rho_O = B_H(0,0)$. Every retained row agrees with B_H at $t+1$ honest indices, hence is a row of B_H, and every honest party retains at least $t+1$ rows and never uses the fallback, so all interpolate to ρ_O. If no honest-consistent k passes, ρ_O is immaterial and is set to the zero vector. □

Lemma 7.6 (Sharing-determined rejection)**.** *For every honest P_h, every proposed assignment k and every opening vector τ,*

$$\mathsf{Dec}_{O,h}(\sigma_h, k; \tau) = \perp \quad \Longleftrightarrow \quad \mathsf{Reject}_O(\Gamma_O, k) = 1.$$

In particular, whether P_h outputs $\perp$ depends only on the sharing transcript and k, neither on the openings nor on h. This holds in every execution, not only on $\mathcal{G}$.

Proof. The rejection test is the only step of $\mathsf{Dec}_{O,h}$ that returns $\perp$: if (G1)–(G4) pass, every retained row has coefficients in $\mathbb{E}^h$, so interpolation returns a vector in $\mathbb{E}^h$, and if fewer than $t+1$ rows are retained the fallback rule also returns a vector in $\mathbb{E}^h$, including when the dealer's fallback is missing or malformed. The tests (G1)–(G4) read the repair declaration, the round-two broadcasts and the EARLY values, which are computed from the round-three candidates and k; they read neither τ nor P_h's private LATE verdicts. □

Proof of Theorem 7.1. The scheme has three sharing rounds, one reconstruction round, and the broadcast-then-decode form with κ_w fixed after round one and of length m_{WS} independent of h. Privacy is Lemma 7.2. On $\mathcal{G}$, which has probability at least $1 - \varepsilon_{\mathrm{WSS}}$ by Lemma 7.3, Lemma 7.4 gives correctness, Lemma 7.5 gives property (a) of Definition 6.2 for every honest-consistent k, and Lemma 7.6 gives property (b). Weak commitment with common abort (Definition 4.1) follows by taking $k = k^{\mathrm{ann}}$, which is honest-consistent because honest parties announce their own vectors. Communication is M IC instances of width h plus $O(n^2)$ vectors of width h and n vectors of length m_{WS}, and all computation is polynomial. □

7.4 Efficient key search

The compiler of Section 8 evaluates Out on assignments in which some parties' keys are not yet known, and needs to decide whether some choice of the unknown keys makes the scheme reject, or accept. We make this a property of the scheme.

Definition 7.7 (Efficient key search)**.** *A WSS of broadcast-then-decode type has* efficient key search *if there is a deterministic polynomial-time algorithm* Search *that, on input a party's state* σ_i*, the openings* τ*, a set* $Z \subseteq [n]$*, keys* $c_j \in \mathbb{E}^\ell \cup \{\mathsf{bad}\}$ *for* $j \notin Z$*, and* $b \in \{\mathsf{rej}, \mathsf{acc}\}$*, returns an assignment* K *with* $k_j = c_j$ *for* $j \notin Z$ *and* $k_j \in \mathbb{E}^\ell \cup \{\mathsf{bad}\}$ *for* $j \in Z$ *such that* $\mathsf{Out}(\sigma_i, K; \tau) = \perp$ *if* $b = \mathsf{rej}$ *and* $\mathsf{Out}(\sigma_i, K; \tau) \neq \perp$ *if* $b = \mathsf{acc}$*, or returns* none *if no such assignment exists. The answer must be exact on every input, and* Search *is not told which parties are honest.*

Exhaustive enumeration implements Search in exponential time, so the definition concerns efficiency only.

For our scheme, $\mathsf{Out} = \perp$ iff $\mathsf{Reject}_O(\Gamma_O, k) = 1$ (Lemma 7.6, on every input), a public predicate that is the conjunction of (G1), which does not involve keys, of one constraint per unordered pair $\{i, j\}$ on the outcomes of S_{ij} and S_{ji} (G2), and of one constraint per label Q_{ij} on its outcome (G3, G4). The outcome of an EARLY label under an assignment that agrees with the fixed keys outside Z is determined by the label's public data once the votes of the free parties are chosen, and a free party can be given a valid key that accepts, $(\alpha, P_L(\alpha), 0)$, or one that rejects, $(\alpha, P_L(\alpha) + 1, T_L(\alpha) - d_L(P_L(\alpha) + 1))$, for any $\alpha \in \mathcal{A}_L$. Hence, with h_0 fixed accepting keys and z free parties, the candidate's value is attainable iff $h_0 + z \geq t + 1$ and $\perp$ iff $h_0 \leq t$, each with an explicit witness.

Lemma 7.8 (Efficient key search)**.** *The scheme of Section 7.2 has efficient key search:* Search *can be implemented in deterministic polynomial time, for every sharing transcript.*

Proof sketch. Compute, for every EARLY label, its attainable outcomes and their slot witnesses as above. Since the constraints of Reject_a involve disjoint labels and the slots of distinct labels vary independently, a rejecting assignment exists iff (G1) fails or some single constraint can be violated by attainable outcomes, and an accepting one iff (G1) holds and every constraint can be satisfied; in both cases the chosen outcomes' witnesses assemble into an assignment with the required fixed coordinates. Appendix A.4 gives the details and a running-time bound. □

As in ordinary WSS, a corrupt party's announcement k_w^{ann} may still change the common rejection decision: sharing fixes ρ_O, not the choice between ρ_O and $\perp$. Removing that choice is the task of the compiler in the following sections.

8 A $(3, 1)$-round VSS from key-uniform WSS

We now turn a key-uniform WSS scheme kuWSS (Definition 6.2) with efficient key search (Definition 7.7) into a VSS scheme with the same round complexity. The construction and its commitment proof use the WSS only through Definition 6.2; the privacy and authentication proofs use the concrete scheme of Section 7. Section 8.1 explains the idea, Section 8.2 gives the protocol, and Section 8.3 proves the VSS properties one by one; longer proofs are in Appendix B.

8.1 Overview

What is missing. In a key-uniform WSS the committed value ρ is fixed by the honest views at the end of sharing, and whether reconstruction returns ρ or $\perp$ is a function of the sharing transcript and the verification keys. The one freedom a corrupt dealer retains is therefore the choice of the corrupt parties' keys, which they announce at reconstruction. VSS requires that this choice too be fixed during sharing.

Masking the keys. Each party P_i collects its verification vectors from all instances into one vector X_i, samples a uniform mask R_i of the same length, shares R_i with its own kuWSS instance, and broadcasts $C_i = X_i + \lambda R_i$ in sharing round two. Once R_i is known, X_i is determined. Since X_i contains P_i's keys for the very instances that share the masks, privacy of the deposits does not follow from key-uniformity alone; Lemma 8.2 proves it for our concrete scheme. The vector X_i includes P_i's keys for the mask instances themselves; this does not regress, because a verification vector has length ℓ independent of the instance's input length, so a mask of length $m = (n+1)\ell$ covers all of a party's keys. Altogether there are $n+1$ instances: the root instance for the dealer's input and one mask instance per party.

Recovering the keys. At reconstruction every party announces its mask, but a corrupt party may lie, and the WSS reconstruction of its mask instance may reject, depending on the very keys being recovered. Two facts from Definition 6.2 resolve this. First, whenever a mask instance decodes under keys that are right on the honest parties, the decoded mask is the committed one, whatever the corrupt keys were; so recovered masks can be cached permanently. Second, a mask instance that can be made to reject by some choice of the still-unknown corrupt keys has a corrupt owner. The recovery therefore alternates between caching masks that decode and closing the set Z of owners whose instances can be made to reject, restarting after every new mask. The final Z is a least fixed point determined by the sharing phase alone, and the final assignment sets the keys of owners in Z to bad and all others to their recovered values. Decoding the root with this assignment yields the VSS output.

Searching in polynomial time. The recovery asks, for an instance and a set Z of free coordinates, whether some choice of the free keys makes the instance reject, or accept. We require this search to be polynomial-time (efficient key search, Definition 7.7); the scheme of Section 7 has this property (Section 7.4), and the recovery then runs in polynomial time with $O(n^3)$ searches.

8.2 Protocol

Notation. Let ℓ be the key length of kuWSS and put $m = (n+1)\ell$. Fix $\lambda \in \mathbb{E} \setminus \{0\}$ whose algebraic degree over $\mathbb{F}$ exceeds nm (Section 8.3 uses this). Instance O_0 has dealer D and input $x \in \mathbb{E}^h$; instance O_a, for $a \in [n]$, has dealer P_a and input a mask $R_a \in \mathbb{E}^m$. Instance O_a has sharing transcript Γ_a, output function Out_a, and committed value ρ_a; $\kappa_i^{(a)} \in \mathbb{E}^\ell$ is P_i's verification vector in O_a, and $\sigma_i^{(a)}$ its state. For a vector $k_i \in \mathbb{E}^m$ let $k_i^{(a)} \in \mathbb{E}^\ell$ be its block for O_a, with $\mathsf{bad}^{(a)} = \mathsf{bad}$. For an assignment $k = (k_1, \ldots, k_n)$, $k_i \in \mathbb{E}^m \cup \{\mathsf{bad}\}$, let

$$D_{a,i}(k) = \mathsf{Out}_a(\sigma_i^{(a)}, (k_j^{(a)})_{j\in[n]}; \tau_a)$$

be P_i's local output in O_a under the keys k and the actual openings τ_a. Let $\mathsf{Key}(C, r) = C - \lambda r$ for $C, r \in \mathbb{E}^m$ and bad otherwise; let $\mathcal{D}_j = \mathbb{E}^m \cup \{\mathsf{bad}\}$ if C_j is well formed and $\{\mathsf{bad}\}$ otherwise; and for $Z \subseteq [n]$ and a vector c let

$$\mathcal{K}(Z, c) = \{k \in \textstyle\prod_j \mathcal{D}_j : \ k_j = c_j \text{ for } j \notin Z\}.$$

Let $B_{\mathrm{bad}} = \{j \in [n] : C_j \text{ is malformed}\}$. $\mathsf{Search}_a(Z, c, b)$ denotes the search algorithm of Definition 7.7 run on instance O_a by P_i with free set $Z \setminus B_{\mathrm{bad}}$, the key fixed to bad for $j \in B_{\mathrm{bad}}$ and to $c_j^{(a)}$ for $j \notin Z \cup B_{\mathrm{bad}}$. A returned assignment $\widetilde{k} = (\widetilde{k}_j)_{j\in[n]}$ is lifted to $\mathcal{K}(Z, c)$ by setting $k_j = c_j$ for $j \notin Z$; $k_j = \mathsf{bad}$ for $j \in Z$ with $j \in B_{\mathrm{bad}}$ or $\widetilde{k}_j = \mathsf{bad}$; and otherwise k_j to the vector with block $\widetilde{k}_j$ in the slot of O_a and zeros in every other block. none is returned as is. Throughout, the WSS is assumed to have efficient key search; exhaustive enumeration is always a (slow) implementation.

Protocol 8: VSS sharing phase

Input. The dealer D holds $x \in \mathbb{E}^h$.

1. **Rounds 1–3.** Run the sharing phase of kuWSS for all $n + 1$ instances concurrently: D shares x in O_0, and each P_a samples $R_a \leftarrow \mathbb{E}^m$ and shares it in O_a.
2. **Round 2, in addition.** Each P_i forms $X_i = (\kappa_i^{(0)}, \ldots, \kappa_i^{(n)}) \in \mathbb{E}^m$ from its round-one verification vectors and broadcasts $C_i = X_i + \lambda R_i$.

Protocol 9: VSS reconstruction phase

Broadcast. Each P_i broadcasts its openings in all $n + 1$ instances and its mask $\widehat{R}_i$ (an honest P_i broadcasts R_i), computed from its saved states. Save the opening vectors $\tau_0, \ldots, \tau_n$; a malformed $\widehat{R}_j$ reads as bad. Verification vectors are not broadcast.
Local computation at P_i.

1. **Initialize.** Let $\mathsf{Cache} = \varnothing$ and $U = [n]$. For each $j \in [n]$ set $k_j^{\mathrm{raw}} = \mathsf{Key}(C_j, \widehat{R}_j)$ and compute $r = D_{j,i}(k^{\mathrm{raw}})$; if $r \neq \perp$, set $\mathsf{Cache}[j] = r$, $c_j = \mathsf{Key}(C_j, r)$, and remove j from U.
2. **Close.** Set $Z = U$. Scan $a \notin Z$ in increasing order; if $\mathsf{Search}_a(Z, c, \mathsf{rej}) \neq \mathsf{none}$, add a to Z and restart the scan. Stop after a scan that adds nothing.
3. **Recover.** Let $U_{\mathrm{old}} = U$. For each $a \in U_{\mathrm{old}}$ in increasing order, let $k = \mathsf{Search}_a(Z, c, \mathsf{acc})$; if $k \neq \mathsf{none}$ and $r = D_{a,i}(k) \neq \perp$, set $\mathsf{Cache}[a] = r$, $c_a = \mathsf{Key}(C_a, r)$, and remove a from U. (Z stays fixed during this step.)
4. **Restart or output.** If $U \neq U_{\mathrm{old}}$, return to step 2. Otherwise set $k_j^\star = \mathsf{bad}$ for $j \in Z$ and $k_j^\star = c_j$ for $j \notin Z$, and output $D_{0,i}(k^\star)$.

Step 4 is well defined because $U \subseteq Z$ throughout: every coordinate outside Z is cached. The number of searches is $O(n^3)$: each restart follows a strict growth of the cache, so there are at most $n + 1$ passes, each with at most n closure additions of n searches and n recovery searches.

8.3 Security

Theorem 8.1 (Main). *Let $t < n/2$. Against a static rushing adversary, the protocol of Section 8.2, instantiated with the key-uniform WSS of Section 7 and the parameters of Section 8.4, is a VSS scheme (Definition 4.2) with three sharing rounds and one reconstruction round, perfect sharing*

privacy, correctness and commitment error at most $2^{-\kappa}$, and communication and computation polynomial in n, κ and the input length h.

The proof is organized by property. Lemma 8.2 is privacy. Lemma 8.3 provides the event $\mathcal{G}$ on which every instance satisfies Definition 6.2, despite the masked broadcasts. On $\mathcal{G}$, Lemma 8.4 shows that all honest parties decode the root with the same sharing-fixed, honest-consistent assignment; Lemmas 8.5 and 8.6 are correctness and commitment.

Lemma 8.2 (Privacy)**.** *If the dealer is honest, the adversary's view through the end of sharing is identically distributed for every input $x \in \mathbb{E}^h$.*

Proof sketch. A one-time-pad argument for $C_i = X_i + \lambda R_i$ does not apply directly, because X_i contains keys of the mask instances, which are themselves masked. Instead, changing x to $x + \Delta$ is compensated by shifting the root polynomial by $\Delta\, p_{\mathcal{C}}(X)p_{\mathcal{C}}(Y)$ and every honest mask R_i by some u_i, with the row, pad and IC translations of Lemma 7.2 applied in every honest-owned instance. The honest keys then move by an $\mathbb{F}$-linear function of (Δ, u), and keeping every C_i fixed is a linear system $(\lambda I + A)u = -b\Delta$ with A over $\mathbb{F}$, which is solvable because λ has degree $> nm$ over $\mathbb{F}$. The resulting map is a bijection of the honest coins that fixes the adversary's view. Appendix B.1 gives the full proof. □

Lemma 8.3 (Authentication in the composition)**.** *There is an event $\mathcal{G}$ with $\Pr[\neg\mathcal{G}] \le \sum_{O,L} n(\frac{1}{q-1} + \frac{\delta_L}{|\mathcal{A}_L|})$, the sum over all labels of all $n+1$ instances, on which every instance satisfies properties (a) and (b) of Definition 6.2 for every honest-consistent assignment, with the actual openings.*

Proof sketch. The only difference from the standalone setting is that honest keys are broadcast in masked form during sharing. The same coupling as in Lemma 8.2, applied to a single key instead of to the input, shows that the point of each honest key of an exported label is uniform given the adversary's sharing view (Lemma B.1), and kept keys are never broadcast (Lemma B.2). The good event of Lemma 7.3 therefore has the stated probability in every instance, and Theorem 7.1 applies to each. Appendix B.1 gives the details. □

Fix an execution in $\mathcal{G}$. Let $c_j = \mathsf{Key}(C_j, \rho_j)$ be the *canonical* vector, fixed by the honest views; for honest P_j, $c_j = X_j$. Call k honest-consistent if $k_j = X_j$ for all honest P_j.

Lemma 8.4 (Recovery)**.** *On $\mathcal{G}$, every honest party obtains the same assignment $k^\star$, which is honest-consistent and determined by the honest parties' views at the end of sharing.*

Proof. Invariants. Throughout, every cached value is ρ_a for its instance, and $U \subseteq Z \subseteq \mathcal{C}$. Initially k^{raw} is honest-consistent, so by correctness every honest owner's instance decodes to $R_a = \rho_a$ and by (a) every finite result is ρ_a; hence $U \subseteq \mathcal{C}$. While the invariants hold, every honest coordinate outside Z is cached and equals X_j, and only coordinates inside $Z \subseteq \mathcal{C}$ vary, so every $k \in \mathcal{K}(Z, c)$ is honest-consistent. By correctness no honest owner's instance rejects under such a k, so closing adds only corrupt owners; and by (a) a finite $D_{a,i}(k)$ equals ρ_a. Removing a recovered owner from U keeps $U \subseteq Z$. Different honest parties may obtain different witnesses from Search_a, which takes their private states as input; the argument uses only whether a witness exists, which is common to honest parties by (a) and (b), and that every witness decodes to ρ_a.

Fixed point. For the full canonical vector c let $\Phi(Z) = Z \cup \{a : \exists k \in \mathcal{K}(Z, c),\ D_{a,i}(k) = \perp\}$. By (b) the test $D_{a,i}(k) = \perp$ does not depend on the corrupt openings, and by (a) it is the same at every

honest P_i, so Φ is determined by the honest views. Since $c_j \in \mathcal{D}_j$, enlarging Z enlarges $\mathcal{K}(Z, c)$, so Φ is extensive and monotone; let Z_0 be its least fixed point above $\varnothing$. Because all coordinates outside Z are cached and canonical, step 2 computes the least fixed point $\mathrm{Cl}(U)$ above its seed. In the terminal pass no $a \in U$ has an accepting search, so every $k \in \mathcal{K}(Z, c)$, in particular c, rejects O_a; hence $U \subseteq \Phi(\varnothing) \subseteq Z_0$ and $\mathrm{Cl}(U) \subseteq Z_0$. Conversely $\mathrm{Cl}(U)$ is a fixed point containing the least one. So the terminal set is Z_0 at every honest party, and $k^\star$ is bad on Z_0 and c elsewhere: determined by the honest views, the same everywhere, and honest-consistent since $Z_0 \subseteq \mathcal{C}$. □

Lemma 8.5 (Correctness)**.** *On $\mathcal{G}$, if the dealer is honest, every honest party outputs x.*

Proof. $k^\star$ is honest-consistent by Lemma 8.4, so correctness of O_0 (Definition 6.2) gives $D_{0,i}(k^\star) = x$ at every honest P_i. □

Lemma 8.6 (Commitment)**.** *On $\mathcal{G}$, there is a value $s^* \in \mathbb{E}^h \cup \{\perp\}$ determined by the honest parties' views at the end of sharing such that every honest party outputs s^*.*

Proof. Let $s^* = \perp$ if $D_{0,i}(k^\star) = \perp$ and $s^* = \rho_0$ otherwise. By Lemma 8.4, $k^\star$ is common and fixed by the honest views; by (a) all honest parties obtain the same member of $\{\rho_0, \perp\}$ under $k^\star$, and by (b) which member does not depend on the corrupt openings. Hence s^* is determined by the honest views and is every honest party's output. □

Theorem 8.7 (Conditional compiler)**.** *Let* kuWSS *be a WSS of broadcast-then-decode type with three sharing rounds, one reconstruction round, key length independent of the input length, and efficient key search. Consider the protocol of Section 8.2 built from* kuWSS*, and suppose that in this composed execution there is an event $\mathcal{G}$ of probability at least $1 - \varepsilon$ on which all $n + 1$ instances satisfy properties (a) and (b) of Definition 6.2 for every honest-consistent assignment. Then the protocol satisfies correctness and commitment of Definition 4.2 with error at most ε, has three sharing rounds and one reconstruction round, and its local computation is polynomial. If in addition the composed sharing phase is perfectly private, it is a VSS scheme.*

Proof. Lemmas 8.4, 8.5 and 8.6 use the WSS only through (a) and (b) on $\mathcal{G}$ and through Definition 7.7; the round and computation counts are as in the proof of Theorem 8.1 below. □

Proof of Theorem 8.1. Theorem 8.7 applies with $\mathcal{G}$ from Lemma 8.3 and privacy from Lemma 8.2. The sharing phase is the three rounds of the $n + 1$ instances, with one extra broadcast in round two; reconstruction is one broadcast round followed by local computation. Privacy is Lemma 8.2. With the parameters of Section 8.4, Lemma 8.3 gives $\Pr[\neg\mathcal{G}] \le 2^{-\kappa}$, and on $\mathcal{G}$ Lemmas 8.5 and 8.6 give correctness and commitment. Communication consists of one WSS instance of input width h, n mask instances of input width $m = O(n^3)$, and n broadcasts of length m; Section 8.4 bounds the resulting bit complexity. Local reconstruction makes $O(n^3)$ searches, polynomial by Definition 7.7, and $O(n^2)$ evaluations of Out, and is polynomial in n, h and κ. □

8.4 Parameters

The concrete WSS exports keys for $2(n-1)(n-2)$ labels of three field elements each, so $\ell = 6(n-1)(n-2)$ and $m = (n+1)\ell = O(n^3)$. Let $N = (n+1)M < 4(n+1)n^2$ be the total number of labels, and put $h_{\max} = \max\{h, m\}$ and $\delta_{\max} = h_{\max} + t - 1$, the largest degree bound of any label.

Choose the base field $\mathbb{F}$ with $|\mathbb{F}| > 2(h_{\max}+1)$ and $|\mathbb{F}| \geq 2^{\kappa+1}Nn(\delta_{\max}+1)$. Every label then has degree bound at most $\delta_{\max}$ and a checking-point domain of size at least $|\mathbb{F}| - h_{\max} - 1 > |\mathbb{F}|/2$, so

$$Nn\Big(\frac{1}{|\mathbb{F}|-1} + \frac{\delta_{\max}}{|\mathbb{F}|-h_{\max}-1}\Big) \leq 2^{-\kappa}, \tag{8}$$

and $\log_2 |\mathbb{F}| = O(\kappa + \log n + \log(h+1))$. Let $\mathbb{E} = \mathbb{F}[\lambda]$ with λ of algebraic degree $nm+1 = O(n^4)$ over $\mathbb{F}$; an element of $\mathbb{E}$ has $O(n^4(\kappa + \log n + \log(h+1)))$ bits. A WSS instance with input width w communicates $O(n^2(w+n))$ elements of $\mathbb{E}$, counting each broadcast once, so the root instance, the n mask instances and the masked broadcasts communicate $O(n^2h + n^6)$ elements of $\mathbb{E}$, that is $O((n^6h + n^{10})(\kappa + \log n + \log(h+1)))$ bits; for $h = O(n^3)$ this is $O(n^{10}(\kappa + \log n))$ bits. Local computation is polynomial in n, h and κ.

9 AI Methodology

The authors used ChatGPT 6 Astra when developing the key ideas of the construction. The authors then identified the abstractions that are used in this version of the write-up, verified, modified and significantly simplified the original ChatGPT-aided generated ideas. We used ChatGPT 6 Astra and Claude Fable 5.1 to develop the formal proofs (in some cases from scratch, and in other cases, based on our drafts). For writing and proof-checking throughout the paper, we used ChatGPT 6 Astra, Claude Fable 5.1 and Opus 5.5. The authors take full responsibility for the correctness, originality, and integrity of all content, including material prepared with LLM assistance.

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 152–174. Springer, Cham, November 2018.

[ABT19] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 504–531. Springer, Cham, May 2019.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424. Springer, Cham, August 2018.

[AKP20a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 562–594. Springer, Cham, November 2020.

[AKP20b] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st FOCS*, pages 1277–1284. IEEE Computer Society Press, November 2020.

[AKP23] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of statistical MPC with optimal resiliency. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1527–1536. ACM Press, June 2023.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[BKP11] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 590–609. Springer, Berlin, Heidelberg, December 2011.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 311–326. Springer, Berlin, Heidelberg, May 1999.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th FOCS*, pages 383–395. IEEE Computer Society Press, October 1985.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987.

[FGG+06] Matthias Fitzi, Juan A. Garay, Shyamnath Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 329–342. Springer, Berlin, Heidelberg, March 2006.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589. ACM Press, July 2001.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: Information-theoretic and black-box. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 123–151. Springer, Cham, November 2018.

[KKK08] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 499–510. Springer, Berlin, Heidelberg, July 2008.

[KPR10] Ranjit Kumaresan, Arpita Patra, and C. Pandu Rangan. The round complexity of verifiable secret sharing: The statistical case. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 431–447. Springer, Berlin, Heidelberg, December 2010.

[PCRR09] Arpita Patra, Ashish Choudhary, Tal Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing revisited. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 487–504. Springer, Berlin, Heidelberg, August 2009.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Berlin, Heidelberg, August 1992.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

A Proofs for Section 7

A.1 Authentication

Realization notation. In the realization of Section 5.1, for a width-h label L the sender state consists of polynomials F_L, G_L of degree at most $h+t-1$; F_L encodes the payload at public nodes $\beta_1, \ldots, \beta_h$. The holder's checkpoint is (d_L, T_L) with $T_L = d_L F_L + G_L$ when it follows the protocol. An original verifier key is $(\alpha_{L,w}, F_L(\alpha_{L,w}), G_L(\alpha_{L,w}))$, with $\alpha_{L,w}$ sampled from $\mathcal{A}_L = \mathbb{F} \setminus \{0, \beta_1, \ldots, \beta_h\}$. Write $\delta_L = h + t - 1$ for the degree bound, P_L for an arbitrary candidate witness, and ℓ_r for the Lagrange basis polynomial at packing node β_r. The prescribed witness is F_L.

Lemma A.1 (Checking-point hiding in the standalone execution)**.** *Fix a label L with an honest sender and a corrupt holder, and an honest verifier P_w. Write*

$$K_{L,w} = (\alpha_{L,w}, F_L(\alpha_{L,w}), G_L(\alpha_{L,w})).$$

For an EARLY label, let V_L be the adversary's complete view through sharing. For a LATE label, let V_L be its complete view at the time its actual reconstruction candidate is fixed, including all honest reconstruction broadcasts available to a rushing adversary. The view includes the adversary's tape. Then, for every positive-probability conditioning event and every $a \in \mathcal{A}_L$,

$$\Pr[\alpha_{L,w} = a \mid V_L = v, F_L = f, G_L = g] = \frac{1}{|\mathcal{A}_L|}. \tag{9}$$

The assertion holds for either an honest or a corrupt instance owner.

Proof. Fix F_L, G_L and all coins except $\alpha_{L,w}$. Changing the point from a to b and updating its private tuple consistently preserves every message visible to a corrupt party through the indicated time. The sender's finish tests the full polynomial identity; sharing flags and opening schedules do not inspect the tuple. An EARLY point is broadcast only after its candidate is fixed, so the sharing view is unchanged.

A LATE point is never broadcast and occurs in no EARLY verification vector. Honest reconstruction messages contain the prescribed holder polynomials, the owner's fallback, and EARLY verification vectors, none of which uses the target LATE point, even when the verifier is also the sender. Thus the switch preserves every honest message visible before a rushing corrupt holder fixes its LATE candidate. The assertion concerns this view, before any verification verdict is exposed. The switch preserves the sender polynomials and all other random coins, has the reverse switch as inverse, and relates equal-probability fibers, proving (9). □

For a label of degree bound δ_L, an honest sender that continues has $T_L = d_L F_L + G_L$. Its original tuple makes the discrepancy disjunct false. Hence a valid actual candidate $P_L \neq F_L$ can be accepted by an honest verifier only if $(P_L - F_L)(\alpha_{L,w}) = 0$. Conditional on the view in Lemma A.1, the candidate and these polynomials are fixed, so the probability is at most $\delta_L / |\mathcal{A}_L|$. A sender replacement gives the honest sender's true payload directly. A malformed candidate is rejected.

For an honest holder with a valid received polynomial, its polynomials and each original normalized tuple are fixed before its fresh challenge $d_L \leftarrow \mathbb{F}^\times$. If $F_L(\alpha) \neq y$, rejection of that holder's candidate requires

$$d_L(F_L(\alpha) - y) = z - G_L(\alpha),$$

which specifies at most one challenge. Its probability is at most $1/(|\mathbb{F}|-1)$. This bound also holds for a corrupt sender, without assuming checking-point hiding. Tuples are normalized in round one and cannot be changed after the challenge.

Proof of Lemma 7.3. For (ii): an honest sender with $c_L = 0$ has $T_L = d_L F_L + G_L$, so at its original key the second disjunct of Open is false, and a valid candidate $P_L \neq F_L$ is accepted only if $(P_L - F_L)(\alpha_{L,w}) = 0$. Conditional on the view of Lemma A.1, the candidate and the polynomials are fixed while $\alpha_{L,w}$ is uniform on $\mathcal{A}_L$, so this has probability at most $\delta_L/|\mathcal{A}_L|$. For (i): the holder's polynomials and each stored key are fixed before its fresh $d_L \leftarrow \mathbb{F}^\times$; if $F_L(\alpha) \neq y$, rejection of its candidate requires $d_L(F_L(\alpha) - y) = z - G_L(\alpha)$, one value of d_L, probability at most $1/(q-1)$; this holds for a corrupt sender and a malformed key. A union bound over M labels and n verifiers gives the bound. The vote counts follow because at least $n - t$ parties are honest and at most t are corrupt. □

A.2 The row argument

Fix an execution in $\mathcal{G}$ and an honest-consistent k. We first record that honest holders always have their candidates present.

Lemma A.2 (Readiness and endpoint guarantees)**.** *Every required EARLY label with an honest holder has its prescribed candidate slot present. In the absence of a valid replacement, an honest holder with a valid payload opens that payload at every honest party, and in EARLY mode under every honest-consistent key assignment, whenever the prescribed opening is scheduled. For an honest sender, every finite opening result equals its original payload.*

Proof. For S_{ij}, every required condition is public after round two or is a replacement issued by P_i itself as sender. For Q_{ij}, an honest holder is the owner and knows its own repair set. Thus its sending rule agrees with the required set. For a valid honest-holder opening at least $n - t \geq t + 1$ honest votes accept. For a false honest-sender opening every honest vote rejects, leaving at most t accepting proposed keys. Replacements from honest senders contain the true value. The private LATE claims follow directly from the IC guarantees. □

Invalid received payloads have explicit consequences: a bad W or S payload raises the row flag; a bad incoming R payload suppresses its comparison and uncorrected opening; a bad Q payload at an honest owner makes $a^D_{ij} = \perp$, forcing $i \in U$, so that Q_{ij} is not required.

Lemma A.3 (Retention)**.** *If the dealer is corrupt and* $\mathsf{Reject}_O(\Gamma_O, k) = 0$*, every honest row is retained by every honest party with its prefix row.*

Proof. Repaired rows are retained by definition. Let P_i be honest with $i \notin U$. G1 implies $\mathrm{inv}_i = 0$ and no row replacement. Lemma A.2 makes every coefficient and every activated S_{ij} equal its valid local value, so the first two retention conditions hold with $f_i = g_i$.

For a repaired neighbor P_j, if P_i replaces Q_{ij}, the required S_{ij} is present and gives $g_i(j)$; G2 equates it to $u_j(i)$. Otherwise Q_{ij} is required and finite by G3, and sender authenticity gives $q_{ij} = r_{ij}$. Since $\eta_i = 0$, G4 yields

$$g_i(j) + r_{ij} = a_{ij} = a^D_{ij} = u_j(i) + q_{ij},$$

which gives the same row equality. Every finite outgoing pad is r_{ij}, so its test holds. For an uncorrected incoming pad with $\phi_{ij} = 0$, holder P_i reliably opens exactly the value used in b_{ij}. Thus every neighbour condition holds, and row i is retained. □

Lemma A.4 (Reciprocity)**.** *Under the hypotheses of Lemma A.3, every row i retained by an honest party and every honest row $j \neq i$ satisfy $f_i(j) = f_j(i)$.*

Proof. If both rows are repaired, G2 gives the equality. If exactly one is repaired, the unrepaired row's repaired-neighbor filter gives it. Suppose both are unrepaired. A failed cross comparison $a_{ij} = b_{ji}$ or $a_{ji} = b_{ij}$ activates both S labels. Replacements on both R_{ij}, R_{ji} also activate both. Retention and G2 then equate the row entries.

Otherwise both cross comparisons hold and at least one pad direction is uncorrected. By the receiver-side normalization of sharing round two, $b_{ij} = \perp$ whenever $\phi_{ij} = 1$, and a comparison with a $\perp$ operand fails; hence $a_{ji} = b_{ij}$ gives $\phi_{ij} = 0$, and $a_{ij} = b_{ji}$ gives $\phi_{ji} = 0$. If R_{ji} is uncorrected, sender P_j is honest and P_i's incoming test forces a finite $v_{ji} = r_{ji}$. Hence

$$f_i(j) = b_{ij} - r_{ji} = a_{ji} - r_{ji} = f_j(i).$$

If R_{ij} is uncorrected, honest holder P_j reliably opens $v_{ij} = r_{ij}^{(j)}$. The outgoing test of P_i gives

$$f_i(j) = a_{ij} - r_{ij}^{(j)} = b_{ji} - r_{ij}^{(j)} = f_j(i).$$

□

A.3 Privacy

Proof of Lemma 7.2. Fix the corrupt set $\mathcal{C}$ and the adversary's tape. Define

$$p_{\mathcal{C}}(X) = \prod_{c \in \mathcal{C}} \frac{X - a_c}{-a_c}.$$

Then $p_{\mathcal{C}}(0) = 1$, $p_{\mathcal{C}}(a_c) = 0$ for $c \in \mathcal{C}$, and its degree is at most t. To change the input from s to $s + \Delta$, translate the owner's symmetric polynomial by

$$B(X, Y) \longmapsto B(X, Y) + \Delta p_{\mathcal{C}}(X) p_{\mathcal{C}}(Y).$$

Write $e_{ij} = \Delta p_{\mathcal{C}}(a_i) p_{\mathcal{C}}(a_j)$, and translate each honest sender's pad by $r_{ij} \mapsto r_{ij} - e_{ij}$. A pad involving a corrupt index has zero increment. The induced payload increments are e_{ij} for S_{ij}, the corresponding row-coefficient increments for $W_{i,\ell}$, and $-e_{ij}$ for R_{ij}, Q_{ij}. Every label with a nonzero increment has both endpoints honest: the owner is honest, a corrupt row has zero increment, and a pad incident to a corrupt party has zero increment.

For an honest-honest label with payload increment $\xi \in \mathbb{E}^h$, let Γ_L be the set of distinct checking points of corrupt verifiers, so $|\Gamma_L| \leq t$, and set

$$Z_L(X) = \prod_{\gamma \in \Gamma_L} (X - \gamma), \qquad H_{L,\xi}(X) = Z_L(X) \sum_{r=1}^{h} \frac{\xi_r \ell_r(X)}{Z_L(\beta_r)}.$$

This polynomial has degree at most $h + t - 1$, equals ξ_r at β_r, and vanishes at every corrupt verifier's point. Fixing the honest holder's challenge d_L, translate

$$F_L \longmapsto F_L + H_{L,\xi}, \qquad G_L \longmapsto G_L - d_L H_{L,\xi}. \tag{10}$$

This preserves the entire checkpoint $d_L F_L + G_L$ and every corrupt verifier's tuple. Use the identity translation for a zero-increment label, including every label with a corrupt endpoint. All checking points and all honest challenge coins are fixed.

Every round-one message received by the adversary is preserved: rows at corrupt indices, authentication polynomials sent to corrupt holders, and verification tuples sent to corrupt verifiers are unchanged. A corrupt sender's private messages are unchanged because its view and tape are unchanged.

In round two every checkpoint is unchanged, and the increments in the public row comparisons cancel:

$$\delta(g_i(j) + r_{ij}) = e_{ij} - e_{ij} = 0, \qquad \delta(g_i(j) + r_{ji}^{(i)}) = e_{ij} - e_{ji} = 0.$$

The same cancellation holds in the owner's comparisons. Comparisons involving corrupt or malformed data have unchanged row evaluations and corrupt data. All validity and missing-pad flags are preserved, including forced $\perp$ entries.

Every honest round-three disclosure is preserved, without assuming authentication succeeds. An honest row has correct S/W data, and its outgoing Q pads reach the honest owner correctly. Thus $\eta_i = 0$ for every honest P_i, no honest row has an S/W replacement, and the honest owner's repair set satisfies $U \subseteq \mathcal{C}$. All repaired rows have zero increment. An honest sender replaces a label only when its holder is corrupt; the label's whole encoding and payload are unchanged by the translation. An honest holder P_i opens an S_{ij} label only when P_j is corrupt: for two honest indices the comparisons agree, their R_{ij} label does not fail, and none of P_i's Q labels to the honest owner fails. Thus every such disclosed S_{ij} has zero increment, including its full opening polynomial. Finally, the honest owner opens Q_{ij} only for $j \in U \subseteq \mathcal{C}$; these pads and their opening polynomials also have zero increment.

Malformed corrupt-sender data, voluntary corrupt openings, and corrupt replacement messages are also unchanged. Honest sending rules for any resulting repairs or required openings use only the unchanged comparisons and the sender's planned finish. Corrupt round-three actions cannot change simultaneous honest sending decisions. The honest owner sends well-formed repairs, no honest sharing message uses an EARLY verification tuple, and the protocol has no other honest disclosure branches.

For the probability-preserving coupling, fix all checking points and potential honest challenge coins, including those sampled in advance for unused branches. The remaining honest pads and sender polynomials are uniform on affine spaces of fixed size. The map translates these spaces, with inverse given by $-\Delta$. The adversary sees the same honest messages before each rushing choice and therefore makes the same choices. Every corrupt state and message is preserved throughout sharing, even when a later authentication test fails. Averaging over the fixed coins proves the claim. □

A.4 The polynomial search

Fix an instance O of the scheme, $Z \subseteq [n]$ and the fixed keys c. We write $\mathsf{PASS}(O, Z, c)$ and $\mathsf{FAIL}(O, Z, c)$ for Search with $b = \mathsf{acc}$ and $b = \mathsf{rej}$.

Attainable label outcomes. Activation bits, replacements, candidates, and checkpoints are sharing-fixed. For an exported label L, let $\mathcal{K}(Z, c)$ be the set of assignments that agree with c outside Z, and define

$$\mathcal{O}_L(Z) = \{\mathsf{Early}(L; k) : k \in \mathcal{K}(Z, c)\}.$$

The public gates apply in order. A valid replacement gives its value regardless of activation. Without a replacement, an inactive label gives unavailable; an active label with a missing or invalid candidate gives $\perp$; and an active label with a malformed checkpoint gives $\perp$ because every proposed key rejects. Each case has a singleton outcome set.

In the remaining case the label is active, has no replacement, and has a valid fixed candidate polynomial $P_L \in \mathbb{E}[X]$ of degree at most $\delta_L = h_L + t - 1$, with payload $x_L = (P_L(\beta_1), \ldots, P_L(\beta_{h_L}))$. If the opening packet also announces a payload, its agreement with these evaluations is part of candidate validity. Its checkpoint is a well-formed pair (d_L, T_L), where $d_L \in \mathbb{F}^\times$ and $\deg T_L \le \delta_L$. For a valid proposed key $K = (\alpha, y, z) \in \mathcal{A}_L \times \mathbb{E} \times \mathbb{E}$, its vote is

$$\mathsf{Check}_L(K) = [P_L(\alpha) = y] \ \vee \ [T_L(\alpha) \neq d_L y + z].$$

Here $\mathcal{A}_L \subseteq \mathbb{F}$ is the nonempty public checking-point domain, disjoint from the packing nodes and from zero.

Let

$$z = |Z|, \qquad h_0 = \sum_{w \notin Z} \mathsf{Check}_L(c_w[L]).$$

Every coordinate in Z is free, and a fixed coordinate equal to bad contributes no vote. (In the VSS, coordinates with malformed deposits are fixed to bad by the wrapper of Section 8.2 and are not in Z.) All n parties, including the sender and holder, are counted in the threshold $t + 1$.

Lemma A.5 (Exact attainable outcomes)**.** *In this remaining case,*

$$x_L \in \mathcal{O}_L(Z) \iff h_0 + z \ge t + 1, \qquad \perp \in \mathcal{O}_L(Z) \iff h_0 \le t.$$

Every attainable outcome has an explicitly constructible assignment of the free key slots of L.

Proof. Choose any $\alpha \in \mathcal{A}_L$ and put $p = P_L(\alpha)$ and $q = T_L(\alpha)$. The valid keys

$$K_L^+ = (\alpha, p, 0), \qquad K_L^- = (\alpha, p + 1, q - d_L(p + 1))$$

have accepting and rejecting votes, respectively. For K_L^+ the first disjunct holds. For K_L^- the first disjunct fails because $1 \neq 0$ in $\mathbb{E}$, and the second fails by the definition of its third coordinate. These keys are valid in every field characteristic and for packed payloads: the checking point and challenge lie in $\mathbb{F}$, while p, q, y, z lie in $\mathbb{E}$.

Each party P_w with $w \in Z$ can independently use either key in this slot. Thus the possible vote counts are precisely the integers from h_0 to $h_0 + z$. To realize x_L, give an accepting key to $\max\{0, t + 1 - h_0\}$ free parties and a rejecting key to the rest. To realize $\perp$, give every free party a rejecting key. These choices are available exactly under the displayed inequalities. They can be extended to complete vectors in $\mathcal{K}(Z, c)$ by filling all other free slots arbitrarily. □

Disjoint constraint groups. The condition $\mathsf{Reject}_O(\Gamma_O, k) = 0$ is the conjunction of the following constraints.

1. The key-independent repair checks (G1).
2. For each unordered pair $\{i, j\}$, the single (G2) constraint: the finite values among s_{ij}, s_{ji} agree. If $i \in U$, the value $s_{ij} = u_i(j)$ is fixed and does not depend on the S_{ij} key slot; otherwise $s_{ij} = \mathsf{Early}(S_{ij}; k)$. This group uses at most the two labels S_{ij} and S_{ji}.

3. For each Q_{ij}, one group combining *all* its applicable (G3) and (G4) conditions. If it is activated, its outcome must be a finite payload. If $i \notin U$, $j \in U$, and Q_{ij} is unreplaced, the same outcome must also satisfy
$$a_{ij}^D = b_{ji}^D = u_j(i) + \mathsf{Early}(Q_{ij}; k).$$
An unsolicited holder opening activates the label even when malformed; in that case an invalid candidate gives the singleton outcome $\perp$, and the finite-outcome requirement fails. Missing or malformed public operands in a required equality make that equality false.

Conditions (G3) and (G4) use the same outcome of Q_{ij} and belong to one group. Each Q label occurs only in that group, and each S label only in its unordered-pair group. Row-decoder tests, including tests of activated S labels, are not constraints of Reject_O.

Proof of Lemma 7.8. By Lemma 7.6, on every input $\mathsf{Out} = \perp$ iff $\mathsf{Reject}_O(\Gamma_O, K) = 1$, so searching for a rejecting or accepting assignment is exactly searching for an assignment that fails or passes the public predicate, also outside the event $\mathcal{G}$.

First test (G1). If it fails, PASS returns none and FAIL returns an arbitrary assignment, without evaluating the remaining groups. Otherwise compute every relevant $\mathcal{O}_L(Z)$ and its slot witnesses. Each set has size at most two. Test a pair group on at most four pairs of outcomes, and a Q group on at most two outcomes, incorporating all public gates and all conditions of that group.

For PASS, return none if some group has no satisfying outcome pattern. Otherwise choose one satisfying pattern for each group. For FAIL, a witness exists precisely when some group has a violating pattern; choose such a pattern and leave all other groups unconstrained. If no such pattern exists, return none.

To construct either witness, start with the arbitrary assignment described above and insert the slot witnesses for the chosen patterns. Groups use disjoint labels, whose slots vary independently in each free field vector, so the choices are compatible. Fixed coordinates, including those equal to bad, are unchanged. Singleton public outcomes need no slot modification. The resulting full assignment belongs to $\mathcal{K}(Z, c)$ and satisfies the requested predicate. Conversely, every full assignment induces one of the enumerated patterns in every group, proving both negative answers as well. □

Running time. An instance has $O(n^2)$ exported labels. Validating candidates and computing the fixed-key vote counts costs $O(n^3(\delta + 1))$ field operations for degree bound δ, and writing a witness costs $O(nm)$ entries. For mask instances $\delta = O(n^3)$, so all $O(n^3)$ searches of the recovery use $O(n^{11})$ operations in $\mathbb{E}$.

B Proofs for Section 8

B.1 Privacy and hidden points in the composition

Throughout, the *deposit* of P_w is its masked broadcast $C_w = X_w + \lambda R_w$ of sharing round two. Fix the static corrupt set $\mathcal{C}$, let $\mathcal{H} = [n] \setminus \mathcal{C}$, and put
$$p_{\mathcal{C}}(X) = \prod_{c \in \mathcal{C}} \frac{X - a_c}{-a_c}.$$

Thus $p_{\mathcal{C}}(0) = 1$, $p_{\mathcal{C}}(a_c) = 0$ for $c \in \mathcal{C}$, and $\deg p_{\mathcal{C}} \le t$. All party indices, packing nodes, checking points, and challenges belong to $\mathbb{F}$; payloads and masks belong to $\mathbb{E}$. Choose $\lambda \in \mathbb{E}$ of algebraic degree $L > nm$ over $\mathbb{F}$, where each mask has width m.

A translation of packed authentication data. Consider an honest-sender, honest-holder label of width h and degree bound $h + t - 1$. Write d_L for its challenge, and let Γ_L be the set of distinct checking points sent to corrupt verifiers. There are at most t such points, none a packing node. For a payload increment $\xi = (\xi_1, \dots, \xi_h) \in \mathbb{E}^h$, define

$$Z_L(X) = \prod_{\gamma \in \Gamma_L} (X - \gamma), \qquad H_{L,\xi}(X) = Z_L(X) \sum_{r=1}^{h} \frac{\xi_r \ell_r(X)}{Z_L(\beta_r)},$$

where ℓ_r is the Lagrange basis for the packing nodes. This polynomial has degree at most $h + t - 1$, takes value ξ_r at β_r, and vanishes on Γ_L. The translation

$$F_L \longmapsto F_L + H_{L,\xi}, \qquad G_L \longmapsto G_L - d_L H_{L,\xi} \tag{11}$$

preserves the public checkpoint $T_L = d_L F_L + G_L$ and every corrupt verifier's tuple. An honest verifier's tuple changes by

$$(0, H_{L,\xi}(\alpha), -d_L H_{L,\xi}(\alpha)).$$

The coefficients of this change as a linear function of ξ lie in $\mathbb{F}$. The translation is a bijection between the uniform polynomial distributions for the two payloads. For $\xi = 0$, it is the identity on the full polynomials.

Proof of Lemma 8.2. The adversary's view includes all public messages, the messages received by corrupt parties, and their local states and random tapes; the argument does not condition on authentication succeeding. Fix the adversary's random tape and compare root secrets s and $s + \Delta$. The root instance is O_0 and the mask instance owned by P_w is O_w. Change the root polynomial by

$$B_0(X, Y) \longmapsto B_0(X, Y) + \Delta p_{\mathcal{C}}(X) p_{\mathcal{C}}(Y).$$

For every honest mask owner P_w, introduce a vector $u_w \in \mathbb{E}^m$, to be chosen below, and change

$$R_w \longmapsto R_w + u_w, \qquad B_w(X, Y) \longmapsto B_w(X, Y) + u_w p_{\mathcal{C}}(X) p_{\mathcal{C}}(Y).$$

All corrupt-owned instances are left unchanged. For an honest-owned instance O, write δ_O for its secret increment (Δ or u_w), and set

$$e_{O,ij} = \delta_O p_{\mathcal{C}}(a_i) p_{\mathcal{C}}(a_j).$$

For each honest pad sender P_i in this instance, translate its pad by $r_{O,ij} \mapsto r_{O,ij} - e_{O,ij}$, specifying the payload increments for its R_{ij} and Q_{ij} labels. The S_{ij} payload increment is $e_{O,ij}$; the $W_{i\ell}$ payload increment is the corresponding coefficient of $\delta_O p_{\mathcal{C}}(a_i) p_{\mathcal{C}}(Y)$.

Every nonzero label increment has two honest endpoints. For S and W, a corrupt holder has index in $\mathcal{C}$ and hence zero row increment. For R, a corrupt sender or holder gives $e_{O,ij} = 0$. For Q, a nonzero increment is possible only in an honest-owned instance with an honest pad sender, so its holder, the instance owner, is also honest. Apply (11) to these honest-honest labels. All labels in corrupt-owned instances, and all labels with a corrupt endpoint, are left unchanged. All checking points and honest challenges are fixed by this map.

Every round-one message received by a corrupt party is unchanged: rows at corrupt indices are fixed, and (11) fixes corrupt verification tuples for honest-honest labels. Checkpoints are also unchanged. In honest-owned instances, the increments in each honest masked comparison cancel; for example,

$$\delta(g_i(j) + r_{ij}) = e_{O,ij} - e_{O,ij} = 0, \qquad \delta(g_i(j) + r_{ji}^{(i)}) = e_{O,ij} - e_{O,ji} = 0.$$

The same cancellation holds for the owner's comparisons. Comparisons in corrupt-owned instances remain unchanged. Validity and missing-pad flags are preserved because corrupt-sender data are fixed and the increments to an honest row's S and W payloads are consistent.

It remains to preserve the honest broadcasts C_w. Stack the u_w, $w \in \mathcal{H}$, into $u \in \mathbb{E}^r$, where $r = |\mathcal{H}|m \le nm$. Since all the authentication increments have coefficients in $\mathbb{F}$, the induced change in the honest deposit vectors has the form

$$\delta X_{\mathcal{H}} = Au + b_\Delta, \qquad A \in \mathbb{F}^{r\times r}.$$

Checking-point coordinates have zero increment; evaluation coordinates have the increments above. The matrix and the linear map $\Delta \mapsto b_\Delta$ depend only on fixed points, challenges, and instance structure, not on translated polynomial coefficients or masks. Preserving all honest C_w is equivalent to

$$(\lambda I_r + A)u = -b_\Delta. \tag{12}$$

The polynomial $\det(ZI_r + A) \in \mathbb{F}[Z]$ is monic of degree $r < L$, so it cannot vanish at λ. Thus (12) has a unique solution. Corrupt round-two messages are unchanged because the adversary's preceding view, including honest broadcasts available to it before sending, is fixed.

For round-three disclosures in an honest-owned instance, the public repair set satisfies $U \subseteq \mathcal{C}$: honest rows are well formed, their outgoing pad copies at the honest owner are correct, and their S/W distributions have honest endpoints and do not fail. Every repaired row therefore has zero increment. An honest sender can fail only a corrupt-holder label, whose payload and authentication polynomials were left unchanged. An honest holder P_i can open S_{ij} only if $j \in \mathcal{C}$: for two honest indices the comparisons agree and R_{ij} does not fail, while none of the honest sender P_i's Q labels to the honest owner fails. Such an S_{ij} has zero increment, so its whole opening polynomial is unchanged. An honest owner opens Q_{ij} only when $j \in U \subseteq \mathcal{C}$, so the pad and its whole opening polynomial have zero increment. All honest disclosures in corrupt-owned instances are unchanged because their honest label data are fixed. The protocol has no sender backups or key-dependent prediction messages.

The same analysis covers malformed messages. Corrupt-sender data and messages are fixed; a missing or invalid incoming pad retains its flag and forced $\perp$ comparison; and a malformed Q copy at an honest owner can force repair only of its corrupt sender's row. Voluntary corrupt openings and surprise corrupt replacements are unchanged. They do not affect another party's simultaneous sending decision, which depends only on earlier comparisons and that sender's planned finish. Every honest owner's repair is well formed, so this accounts for all honest round-three disclosures. Corrupt round-three messages are fixed by the preserved preceding view.

To verify probability preservation, fix all points and all potential honest challenge coins, sampling unused challenge coins in advance if necessary. Sender polynomials, honest pads, and honest mask inputs are uniform on affine spaces of fixed cardinality. The execution map translates these spaces, fixes corrupt coins, and preserves every adversarial choice by preserving its view at each step. Its inverse uses $-\Delta$ and $-u$ in the same equations. It is therefore a probability-preserving bijection that fixes the complete corrupt view, including for malformed corrupt messages and subsequent authentication failures. □

Lemma B.1 (Hidden points of exported labels)**.** *Consider any exported S or Q label L with an honest sender, a corrupt holder, and an honest verifier P_w. Let V_{sh} be the adversary's complete view through the end of sharing, including its tape, its private incoming messages, and every public sharing message. For every supported conditioning event,*

$$\Pr[\alpha_{L,w} = a \mid V_{\mathrm{sh}} = v, \sigma_L = (f, g)] = \frac{1}{|\mathcal{A}_L|} \qquad (a \in \mathcal{A}_L). \tag{13}$$

The statement holds whether or not the root dealer or the owner of the instance containing L is honest. The corrupt holder receives F_L, G_L, so conditioning on them adds no information to its view.

Consequently, for the actual round-three opening polynomial P_L,

$$\Pr\big[\textit{the sender continues}, \quad P_L \neq F_L, \quad \mathsf{Check}_L(\mathsf{val}(P_L), P_L; K_w) = 1\big] \leq \frac{\delta_L}{|\mathcal{A}_L|}.$$

Here malformed openings are rejected, and δ_L and $\mathcal{A}_L$ are the label's degree bound and checking-point domain.

Proof. Fix admissible points a, b. We construct a probability-preserving bijection between executions with $\alpha_{L,w} = a$ and $\alpha_{L,w} = b$ that preserves the adversary's sharing view and the full polynomials F_L, G_L. The direct change to verifier P_w's tuple is

$$v_L = (b - a,\ F_L(b) - F_L(a),\ G_L(b) - G_L(a)).$$

Embed this vector in its slot in the stacked honest deposits, with zero in all other slots, to obtain $v \in \mathbb{E}^r$.

Keep the root instance fixed, change only honest-owned mask instances by $u_w p_{\mathcal{C}}(X) p_{\mathcal{C}}(Y)$, and make the pad and authentication translations from the preceding proof. Corrupt-owned instances remain fixed. The target label has zero payload increment for every u: an S label's corrupt holder has a corrupt row index, and a Q label's corrupt holder owns its untranslated instance. Thus F_L, G_L and the target label's full authentication encoding remain fixed.

Let A be the matrix of changes to honest deposits induced by the mask translations. This matrix is independent of the target point $\alpha_{L,w}$: the target label's increments vanish identically, and all other label points are independently sampled and unchanged. Moreover, $\alpha_{L,w}$ is an honest verifier's random coordinate and is not used to construct any interpolation set of corrupt-verifier points. Those coordinates remain fixed even when their values coincide with the old or new honest point. The target key's rows in A are zero. Solve

$$(\lambda I_r + A)u = -v.$$

As above, $L > nm$ gives invertibility, and the compensating translations preserve every honest masked broadcast C_w.

The round-one corrupt view, checkpoints, and comparisons are preserved as in Lemma 8.2. Every translated mask instance has an honest owner, so the same round-three disclosure analysis applies; the root and corrupt-owned instances are fixed. The direct change to $\alpha_{L,w}$ changes the private tuple delivered to honest P_w, but no adversary-visible honest sharing message except the compensated masked deposit. The sender finish tests the full polynomial identity, and opening schedules do not inspect verifier keys. A malformed checkpoint chosen by the corrupt holder and the sender's resulting replacement are also fixed. Rushing corrupt messages therefore remain unchanged.

The reverse map replaces b by a, with offset $-v$, the same matrix A, and compensating translation $-u$. The maps are inverses and preserve the uniform distributions of all sender coins and masks. Since a, b were arbitrary, conditional uniformity follows. Both evaluation coordinates change consistently with the new point and the fixed defining polynomials.

The holder's actual candidate, including a rushing round-three candidate, is fixed by its preserved view and random tape. If the honest sender continues, its full identity test gives $T_L = d_L F_L + G_L$ and the complaint disjunct is false at every original key. For a well-formed false candidate, acceptance therefore implies $(P_L - F_L)(\alpha_{L,w}) = 0$. This nonzero polynomial has at most δ_L roots. A malformed checkpoint cannot occur on the honest sender's continue branch, and malformed candidates are rejected by the parser. The asserted bound follows. □

Lemma B.2 (Hidden points of kept labels)**.** *Let L be a kept R or W label with an honest sender and a corrupt holder, and let P_w be an honest verifier. Let V_L^{cand} be the adversary's complete view when it fixes its actual reconstruction candidate for L. This view includes its tape, all sharing messages, and all honest reconstruction broadcasts visible to a rushing adversary, including the honest mask announcements. Then*

$$\Pr[\alpha_{L,w} = a \mid V_L^{\mathrm{cand}} = v, \sigma_L = (f, g)] = \frac{1}{|\mathcal{A}_L|} \qquad (a \in \mathcal{A}_L) \tag{14}$$

for every supported conditioning event. In particular, the actual LATE candidate has the same false-acceptance bound $\delta_L/|\mathcal{A}_L|$ as in Lemma B.1.

Proof. Fix all coins except the target point and replace $\Phi_L(a; \sigma_L)$ by $\Phi_L(b; \sigma_L)$ at honest verifier P_w. A kept point and its tuple occur in no deposited vector X_v, so no mask compensation is needed. The point is used only in the party's local verification after all reconstruction broadcasts, not in a checkpoint, sender finish, comparison, repair, holder opening, or owner's fallback. All masked deposits and honest reconstruction broadcasts are therefore fixed, including broadcasts by an honest sender of L. Corrupt messages and the holder's candidate are fixed by its unchanged preceding view and tape.

The switch preserves F_L, G_L and every other coin; its inverse is the reverse switch. Equal-probability fibers give (14), including when the masks disclose EARLY tuples, whose independently sampled points do not depend on the target LATE point. The conditioning excludes later private verification verdicts, and the statement makes no claim after a verdict is published. On the honest sender's continue branch, acceptance of a false candidate requires a root of $P_L - F_L$. □

Honest-holder availability. The masked broadcasts do not affect nonrepudiation. For an honest holder, the original normalized tuple and the holder's polynomials are fixed before its fresh challenge. If the payload polynomial does not match the tuple at its point, rejection requires one prescribed challenge value, giving probability at most $1/(|\mathbb{F}| - 1)$. Later masked broadcasts do not alter this prior tuple choice. The bound does not condition on the complete later view, which contains the challenge, and requires no hiding of a corrupt sender's checking points.

Proof of Lemma 8.3. Apply the honest-holder bound and Lemmas B.1 and B.2 to each candidate and each honest key, and take a union bound; no independence is needed. These are exactly the hypotheses of Lemma 7.3 for every instance, and Theorem 7.1 gives (a) and (b). Hypothetical key assignments change neither an honest key nor an actual candidate, so they require no additional union bound. □